Machine Learning Systems

Principi ed Esercizi di Ingegneria dei Sistemi Artificialmente Intelligenti

Vijay Janapa Reddi

Traduzione italiana: Baldassarre Cesarano

5 novembre 2024

Sommario

Machine Learning Systems offre ai lettori un punto di partenza per comprendere i sistemi di machine learning (ML) basando i concetti sul ML applicato. Poiché le richieste di soluzioni ML efficienti e scalabili aumentano, la capacità di costruire pipeline di ML robuste diventa sempre più cruciale. Questo libro si concentra sulla spiegazione del processo di sviluppo di sistemi ML completi adatti all'implementazione, abbracciando fasi chiave come la raccolta dati, la progettazione del modello, l'ottimizzazione, l'accelerazione, il rafforzamento della sicurezza e l'integrazione, il tutto da una prospettiva di sistema. Il testo copre un'ampia gamma di concetti rilevanti per l'ingegneria ML generale in tutti i settori e le applicazioni, utilizzando TinyML come strumento pedagogico grazie alla sua accessibilità globale. I lettori apprenderanno i principi di base sulla progettazione di architetture di modelli ML, strategie di formazione basate sull'hardware, ottimizzazione delle inferenze performanti e metodologie di benchmarking. Il libro esplora anche considerazioni cruciali sui sistemi in aree come affidabilità, privacy, IA responsabile e validazione delle soluzioni. Buona lettura!

Indice

Prefazione **19**
Perché Abbiamo Scritto Questo Libro **20**
Cosa c'è da Sapere **21**
Dichiarazione di Trasparenza dei Contenuti **22**
Per dare una mano **23**
Contatti **24**
Collaboratori **25**
Copyright **26**
Dedica **27**
Ringraziamenti **28**
Singoli Collaboratori 28
Agenzie e Aziende Finanziatrici 28
Ai Nostri Lettori 28
Collaboratori e Ringraziamenti **29**
Informazioni sul Libro **30**
Panoramica 30
Argomenti Esplorati 30
Chi Dovrebbe Leggerlo 30
Principali Risultati dell'Apprendimento 31
Prerequisiti per i Lettori 31
1. Introduzione **32**
1.1 Panoramica 32
1.2 Cosa c'è nel Libro 35
1.3 Come Orientarsi in Questo Libro 36
1.4 Suddivisione dei Capitoli 37
1.4.1 Nozioni Fondamentali 37
1.4.2 Workflow 37
1.4.3 Training 38
1.4.4 Deployment 38
1.4.5 Argomenti Avanzati 38
1.4.6 Impatto Sociale 39
1.4.7 Chiusura 39
1.5 Contributi dei Lettori 39
2. Sistemi di ML **40**
2.1 Introduzione 41
2.2 Cloud ML 42
2.2.1 Caratteristiche 43
2.2.2 Vantaggi 45
2.2.3 Sfide 46
2.2.4 Casi d'Uso di Esempio 47
2.3 Edge ML 48
2.3.1 Caratteristiche 48
2.3.2 Vantaggi 50
2.3.3 Sfide 50
2.3.4 Casi d'Uso di Esempio 50
2.4 Tiny ML 51
2.4.1 Caratteristiche 51
2.4.2 Vantaggi 53
2.4.3 Sfide 53
2.4.4 Casi d'Uso di Esempio 54
2.5 Confronto 54
2.6 Conclusione 56
2.7 Risorse 56
3. Avvio al Deep Learning **58**
3.1 Introduzione 59
3.1.1 Definizione e Importanza 59
3.1.2 Breve Storia del Deep Learning 59
3.1.3 Applicazioni del Deep Learning 61
3.1.4 Rilevanza per l'IA Embedded 62
3.2 Reti Neurali 62

3.2.1 Perceptron ... 62
3.2.2 Perceptron Multilayer ... 64
3.2.3 Processo di Training ... 65
3.2.4 Architetture dei Modelli ... 66
3.2.5 ML Tradizionale vs Deep Learning ... 69
3.2.6 Scelta tra ML tradizionale e DL ... 70
3.2.7 Fare una Scelta Informata ... 71
3.3 Conclusione ... 72
3.4 Risorse ... 72
4. Workflow dell'IA ... 74
4.1 Panoramica ... 75
4.2 IA Tradizionale o Embedded ... 76
4.2.1 Ottimizzazione delle Risorse ... 77
4.2.2 Elaborazione in Real-time ... 77
4.2.3 Gestione dei Dati e Privacy ... 77
4.2.4 Integrazione Hardware-Software ... 77
4.3 Ruoli e responsabilità ... 77
4.4 Conclusione ... 78
4.5 Risorse ... 79
5. Data Engineering ... 80
5.1 Introduzione ... 81
5.2 Definizione del Problema ... 82
5.3 Ricerca dei Dati ... 86
5.3.1 Dataset preesistenti ... 86
5.3.2 Web Scraping ... 87
5.3.3 Crowdsourcing ... 89
5.3.4 Dati Sintetici ... 90
5.4 Archiviazione Dati ... 92
5.5 Elaborazione dei Dati ... 95
5.6 Etichettatura dei Dati ... 97
5.6.1 Tipi di Etichette ... 97
5.6.2 Metodi di Annotazione ... 99
5.6.3 Garantire la Qualità dell'Etichetta ... 99
5.6.4 Annotazione assistita dall'intelligenza artificiale ... 100
5.7 Controllo della Versione dei Dati ... 101
5.8 Ottimizzazione dei Dati per l'IA Embedded ... 102
5.9 Trasparenza dei Dati ... 103
5.10 Licenze ... 105
5.11 Conclusione ... 107
5.12 Risorse ... 107
6. Framework di IA ... 109
6.1 Introduzione ... 110
6.2 Evoluzione dei Framework ... 111
6.3 Approfondimento su TensorFlow ... 114
6.3.1 Ecosistema TF ... 115
6.3.2 Grafico di Calcolo Statico ... 117
6.3.3 Usabilità & Distribuzione ... 117
6.3.4 Progettazione dell'Architettura ... 118
6.3.5 Funzionalità Native & Keras ... 118
6.3.6 Limitazioni e Sfide ... 119
6.3.7 PyTorch & TensorFlow ... 119
6.4 Componenti di Base del Framework ... 120
6.4.1 Strutture Dati Tensoriali ... 120
PyTorch ... 122
TensorFlow ... 122
6.4.2 Grafi computazionali ... 123
6.4.3 Tool della Pipeline dei Dati ... 128
6.4.4 Data Augmentation ... 128
6.4.5 Funzioni Loss e Algoritmi di Ottimizzazione ... 129
6.4.6 Supporto al Training del Modello ... 130
6.4.7 Validazione e Analisi ... 130
6.4.8 Programmazione differenziabile ... 131
6.4.9 Accelerazione Hardware ... 132

6.5 Funzionalità Avanzate 133
6.5.1 Training distribuito 133
6.5.2 Conversione del Modello 134
6.5.3 AutoML, No-Code/Low-Code ML 134
6.5.4 Metodi di Apprendimento Avanzati 135
6.6 Specializzazione del Framework 137
6.6.1 Cloud 137
6.6.2 Edge 137
6.6.3 Embedded 137
6.7 Framework di IA Embedded 138
6.7.1 Vincoli di Risorse 138
6.7.2 Framework e Librerie 139
6.7.3 Sfide 139
6.8 Esempi 141
6.8.1 Interprete 142
6.8.2 Basati su Compilatore 142
6.8.3 Libreria 143
6.9 Scelta del Framework Giusto 144
6.9.1 Modello 145
6.9.2 Software 145
6.9.3 Hardware 145
6.9.4 Altri Fattori 146
6.10 Tendenze Future nei Framework ML 147
6.10.1 Decomposizione 147
6.10.2 Compilatori e Librerie ad Alte Prestazioni 148
6.10.3 ML per Framework ML 148
6.11 Conclusione 148
6.12 Risorse 149
7. Addestramento dell'IA 150
7.1 Introduzione 151
7.2 Matematica delle Reti Neurali 152
7.2.1 Notazione delle Reti Neurali 153
7.2.2 Funzione Loss come Misura della Bontà di Adattamento Rispetto ai Dati di Addestramento 155
7.2.3 Addestramento di Reti Neurali con Discesa del Gradiente 156
7.2.4 Backpropagation 157
7.3 Grafi del Calcolo Differenziabili 160
7.4 Dati di Training 161
7.4.1 Suddivisioni di Dataset 161
7.4.2 Errori e Insidie Comuni 162
7.5 Algoritmi di Ottimizzazione 168
7.5.1 Ottimizzazioni 168
7.5.2 Compromessi 169
7.5.3 Algoritmi di Benchmarking 169
7.6 Ottimizzazione degli Iperparametri 170
7.6.1 Algoritmi di Ricerca 171
7.6.2 Implicazioni di Sistema 172
7.6.3 Gli Auto Tuner 173
7.7 Regolarizzazione 175
7.7.1 L1 e L2 175
7.7.2 Dropout 177
7.7.3 Arresto Anticipato 177
7.8 Funzioni di Attivazione 178
7.8.1 Sigmoide 179
7.8.2 Tanh 179
7.8.3 ReLU 179
7.8.4 Softmax 180
7.8.5 Pro e Contro 180
7.9 Inizializzazione dei Pesi 181
7.9.1 Inizializzazione Uniforme e Normale 182
7.9.2 Inizializzazione Xavier 182
7.9.3 Inizializzazione He 182
7.10 "Colli di Bottiglia" del Sistema 183
7.10.1 Complessità a Runtime della Moltiplicazione di Matrici 183
7.10.2 Calcolo vs. Collo di Bottiglia della Memoria 185

7.11 Parallelizzazione del Training 188
7.11.1 Parallelismo dei Dati 189
7.11.2 Parallelismo del Modello 190
7.11.3 Confronto 191
7.12 Conclusione 191
7.13 Risorse 192
8. IA Efficiente 194
8.1 Introduzione 195
8.2 La Necessità di un'IA Efficiente 195
8.3 Architetture di Modelli Efficienti 196
8.4 Compressione Efficiente del Modello 196
8.5 Hardware di Inferenza Efficiente 198
8.6 Matematica Efficiente 199
8.6.1 Formati Numerici 200
8.6.2 Vantaggi dell'Efficienza 202
8.7 Valutazione dei Modelli 202
8.7.1 Metriche di Efficienza 203
8.7.2 Confronti di Efficienza 203
8.8 Conclusione 205
8.9 Risorse 206
9. Ottimizzazioni dei Modelli 207
9.1 Introduzione 208
9.2 Rappresentazione Efficiente del Modello 209
9.2.1 Il Pruning 209
9.2.2 Compressione del Modello 219
9.2.3 Modelli Progettati per l'Edge 224
9.3 Rappresentazione Numerica Efficiente 226
9.3.1 Le Basi 227
9.3.2 Vantaggi dell'Efficienza 229
9.3.3 Sfumature della Rappresentazione Numerica 229
9.3.4 Quantizzazione 233
9.3.5 I Tipi 235
9.3.6 Calibrazione 237
9.3.7 Tecniche 241
9.3.8 Pesi vs. Attivazioni 244
9.3.9 Compromessi 245
9.3.10 Quantizzazione e Potatura 246
9.3.11 Quantizzazione Edge-aware 247
9.4 Implementazione Hardware Efficiente 247
9.4.1 Ricerca di Architettura Neurale Basata sull'Hardware 247
9.4.2 Sfide nella "Hardware-Aware Neural Architecture Search" 249
9.4.3 Ottimizzazioni del Kernel 249
9.4.4 Compute-in-Memory (CiM) 250
9.4.5 Ottimizzazione dell'Accesso alla Memoria 250
9.5 Supporto Software e Framework 253
9.5.1 API Native di Ottimizzazione 253
9.5.2 Strumenti di Ottimizzazione Automatizzata 254
9.5.3 Librerie di Ottimizzazione Hardware 255
9.5.4 Visualizzazione delle Ottimizzazioni 255
9.5.5 Conversione e Distribuzione del Modello 258
9.6 Conclusione 259
9.7 Risorse 260
10. Accelerazione IA 262
10.1 Introduzione 263
10.2 Background e Basi 263
10.2.1 Background Storico 263
10.2.2 La Necessità di Accelerazione 264
10.2.3 Principi Generali 265
10.3 Tipi di acceleratori 267
10.3.1 Application-Specific Integrated Circuits (ASIC) 268
10.3.2 Field-Programmable Gate Array (FPGA) 271
10.3.3 Digital Signal Processor (DSP) 273
10.3.4 Graphics Processing Unit (GPU) 275
10.3.5 Central Processing Unit (CPU) 277

10.3.6 Confronto 280
10.4 Co-Progettazione Hardware-Software 280
10.4.1 La Necessità della Progettazione Congiunta 281
10.4.2 Principi di Progettazione Congiunta Hardware-Software 282
10.4.3 Sfide 283
10.5 Software per Hardware AI 284
10.5.1 Modelli di Programmazione 285
10.5.2 Librerie e Runtime 285
10.5.3 Ottimizzazione dei Compilatori 286
10.5.4 Simulazione e Modellazione 286
10.6 Benchmarking dell'Hardware AI 287
10.7 Sfide e Soluzioni 288
10.7.1 Problemi di Portabilità/Compatibilità 289
10.7.2 Problemi di Consumo Energetico 290
10.7.3 Superare i Vincoli delle Risorse 291
10.8 Tecnologie Emergenti 291
10.8.1 Metodi di Integrazione 292
10.8.2 Elaborazione Nùeuromorfica 294
10.8.3 Calcolo Analogico 296
10.8.4 Elettronica Flessibile 297
10.8.5 Tecnologie delle Memorie 299
10.8.6 Calcolo Ottico 300
10.8.7 Quantum Computing 300
10.9 Tendenze Future 303
10.9.1 ML per l'automazione della progettazione hardware 303
10.9.2 Simulazione e Verifica Hardware Basate su ML 304
10.9.3 ML per Architetture Hardware Efficienti 305
10.9.4 ML per Ottimizzare la Produzione e Ridurre i Difetti 305
10.9.5 Verso Modelli di Base per la Progettazione Hardware 306
10.10 Conclusione 307
10.11 Risorse 307
11. Benchmarking dell'IA 309
11.1 Introduzione 310
11.2 Contesto Storico 311
11.2.1 Benchmark delle Prestazioni 311
11.2.2 Benchmark Energetici 312
11.2.3 Benchmark Personalizzati 312
11.2.4 Consenso della Comunità 313
11.3 Benchmark AI: Sistema, Modello e Dati 314
11.3.1 Benchmark di Sistema 314
11.3.2 Benchmark del Modello 314
11.3.3 Benchmark dei Dati 314
11.4 Benchmarking di Sistema 315
11.4.1 Granularità 315
11.4.2 Componenti dei Benchmark 318
11.4.3 Training vs. inferenza 319
11.4.4 I Benchmark del Training 319
11.4.5 Benchmark di Inferenza 324
11.4.6 Misura dell'Efficienza Energetica 328
11.4.7 Esempio di Benchmark 329
11.4.8 Sfide e Limitazioni 330
11.5 Benchmarking del Modello 334
11.5.1 Contesto Storico 335
11.5.2 Metriche del Modello 337
11.5.3 Lezioni Apprese 341
11.5.4 Limitazioni e Sfide 343
11.6 Benchmarking dei Dati 344
11.6.1 Limitazioni dell'IA Incentrata sul Modello 345
11.6.2 Verso un'Intelligenza Artificiale Incentrata sui Dati 346
11.6.3 Benchmarking dei Dati 347
11.6.4 Efficienza dei Dati 348
11.7 La Tripletta 349
11.8 Benchmark per Tecnologie Emergenti 350
11.9 Conclusione 350

11.10 Risorse 351
12. Apprendimento On-Device 352
12.1 Introduzione 352
12.2 Vantaggi e Limiti 353
12.2.1 Vantaggi 354
12.2.2 Limitazioni 356
12.3 Adattamento On-device 358
12.3.1 Riduzione della Complessità del Modello 358
12.3.2 Modifica dei Processi di Ottimizzazione 359
12.3.3 Sviluppo di Nuove Rappresentazioni dei Dati 361
12.4 Il Transfer Learning 362
12.4.1 Specializzazione Pre-Distribuzione 363
12.4.2 Adattamento Post-Distribuzione 364
12.4.3 Vantaggi 365
12.4.4 Concetti Fondamentali 365
12.4.5 Tipi di Apprendimento Tramite Trasferimento 366
12.4.6 Vincoli e Considerazioni 368
12.5 Apprendimento Automatico Federato 369
12.5.1 Panoramica dell'Apprendimento Federato 369
12.5.2 Efficienza della Comunicazione 370
12.5.3 Compressione del Modello 371
12.5.4 Condivisione Selettiva degli Aggiornamenti 371
12.5.5 Aggregazione Ottimizzata 373
12.5.6 Gestione dei Dati non-IID 373
12.5.7 Selezione del Client 374
12.5.8 L'Esempio di Gboard 374
12.5.9 Benchmarking Federated Learning: MedPerf 376
12.6 Problemi di Sicurezza 376
12.6.1 Avvelenamento dei Dati 377
12.6.2 Attacchi Avversari 378
12.6.3 Inversione del Modello 379
12.6.4 Problemi di Sicurezza dell'Apprendimento On-Device 379
12.6.5 Attenuazione dei Rischi dell'Apprendimento On-Device 380
12.6.6 Protezione dei Dati di Training 381
12.7 Framework di Training On-Device 382
12.7.1 Tiny Training Engine 382
12.7.2 Tiny Transfer Learning 383
12.7.3 Tiny Train 384
12.7.4 Confronto 384
12.8 Conclusione 385
12.9 Risorse 386
13. Operazioni di ML 388
13.1 Introduzione 389
13.2 Contesto Storico 390
13.2.1 DevOps 390
13.2.2 MLOps 390
13.3 Componenti Chiave di MLOps 391
13.3.1 Gestione dei Dati 392
13.3.2 Pipeline CI/CD 393
13.3.3 Addestramento del Modello 394
13.3.4 Valutazione del Modello 395
13.3.5 Distribuzione del Modello 395
13.3.6 Model Serving 396
13.3.7 Gestione dell'Infrastruttura 397
13.3.8 Monitoraggio 398
13.3.9 Governance 399
13.3.10 Comunicazione e Collaborazione 399
13.4 Debito Tecnico Nascosto nei Sistemi ML 400
13.4.1 Erosione dei Confini del Modello 400
13.4.2 Intreccio 400
13.4.3 Cascate di Correzione 401
13.4.4 Consumatori Non Dichiarati 401
13.4.5 Debito di Dipendenza dai Dati 402
13.4.6 Debito di Analisi dai Cicli di Feedback 402
13.4.7 Le Giungle di Pipeline 402

13.4.8 Debito di Configurazione 403
13.4.9 Il Mondo che Cambia 403
13.4.10 Gestire il Debito Tecnico nelle Fasi Iniziali 403
13.4.11 Riepilogo 404
13.5 Ruoli e Responsabilità 404
13.5.1 Ingegneri dei Dati 404
13.5.2 Data Scientist 405
13.5.3 ML Engineer 406
13.5.4 DevOps Engineer 406
13.5.5 Project Manager 407
13.6 Sfide dei Sistemi Embedded 407
13.6.1 Risorse di Elaborazione Limitate 408
13.6.2 Memoria Limitata 408
13.6.3 Connettività Intermittente 408
13.6.4 Limitazioni della Potenza 408
13.6.5 Gestione della Flotta 408
13.6.6 Raccolta Dati On-Device 409
13.6.7 Personalizzazione Specifica del Dispositivo 409
13.6.8 Considerazioni sulla Sicurezza 409
13.6.9 Diversi Target Hardware 409
13.6.10 Copertura dei Test 409
13.6.11 Rilevamento della Deriva del Concetto 409
13.7 MLOps Tradizionali e MLOps Embedded 410
13.7.1 Gestione del Ciclo di Vita del Modello 411
13.7.2 Integrazione di Sviluppo e Operazioni 414
13.7.3 Eccellenza operativa 416
13.7.4 Confronto 417
13.7.5 MLOps Tradizionali 417
13.7.6 MLOps Embedded 418
13.8 Casi di Studio 422
13.8.1 Oura Ring 422
13.8.2 ClinAIOps 423
13.9 Conclusione 429
13.10 Risorse 429
14. Sicurezza e Privacy 432
14.1 Introduzione 433
14.2 Terminologia 434
14.3 Precedenti Storici 434
14.3.1 Stuxnet 434
14.3.2 Hack della Jeep Cherokee 435
14.3.3 Botnet Mirai 436
14.3.4 Implicazioni 436
14.4 Minacce alla Sicurezza per i Modelli ML 437
14.4.1 Furto di Modelli 437
14.4.2 Avvelenamento dei Dati 439
14.4.3 Attacchi Avversari 442
14.5 Minacce alla Sicurezza Per l'Hardware ML 444
14.5.1 Bug Hardware 445
14.5.2 Attacchi Fisici 445
14.5.3 Attacchi di Fault-injection 446
14.5.4 Attacchi a canale laterale 448
14.5.5 Interfacce con Perdite 451
14.5.6 Hardware Contraffatto 452
14.5.7 Rischi della Catena di Fornitura 453
14.5.8 Caso di Studio 453
14.6 Sicurezza Hardware del ML Embedded 454
14.6.1 Trusted Execution Environments 454
14.6.2 Avvio Sicuro 457
14.6.3 Moduli di Sicurezza Hardware 460
14.6.4 Physical Unclonable Functions (PUF) 462
14.7 Problemi di Privacy nella Gestione dei Dati 464
14.7.1 Tipi di Dati Sensibili 464
14.7.2 Regolamenti Applicabili 465
14.7.3 De-identificazione 465
14.7.4 Riduzione al Minimo dei Dati 466

14.7.5 Consenso e Trasparenza ... 467
14.7.6 Problemi di Privacy nell'Apprendimento Automatico ... 468
14.8 Tecniche ML per la Tutela della Privacy ... 470
14.8.1 Privacy Differenziale ... 470
14.8.2 Il Federated Learning ... 473
14.8.3 Machine Unlearning ... 477
14.8.4 Crittografia Omomorfica ... 479
14.8.5 Secure Multiparty Communication ... 481
14.8.6 Generazione di Dati Sintetici ... 482
14.8.7 Riepilogo ... 485
14.9 Conclusione ... 486
14.10 Risorse ... 486
15. IA Responsabile ... 488
15.1 Introduzione ... 489
15.2 Definizione ... 489
15.3 Principi e Concetti ... 490
15.3.1 Trasparenza e Spiegabilità ... 490
15.3.2 Equità, Bias [pregiudizi] e Discriminazione ... 490
15.3.3 Privacy e Governance dei Dati ... 490
15.3.4 Sicurezza e Robustezza ... 490
15.3.5 Responsabilità e Governance ... 491
15.4 Cloud, Edge e Tiny ML ... 491
15.4.1 Riepilogo ... 491
15.4.2 Spiegabilità ... 492
15.4.3 Equità ... 492
15.4.4 Sicurezza ... 492
15.4.5 Responsabilità ... 493
15.4.6 Governance ... 493
15.4.7 Privacy ... 493
15.5 Aspetti Tecnici ... 494
15.5.1 Rilevamento e Mitigazione dei Pregiudizi ... 494
15.5.2 Preservare la Privacy ... 496
15.5.3 Machine Unlearning ... 497
15.5.4 Esempi Avversari e Robustezza ... 498
15.5.5 Creazione di Modelli Interpretabili ... 500
15.5.6 Monitoraggio delle Prestazioni del Modello ... 502
15.6 Sfide di Implementazione ... 502
15.6.1 Strutture Organizzative e Culturali ... 502
15.6.2 Ottenere Dati di Qualità e Rappresentativi ... 503
15.6.3 Bilanciamento di Accuratezza e Altri Obiettivi ... 505
15.7 Considerazioni Etiche Nella Progettazione dell'IA ... 505
15.7.1 Sicurezza dell'Intelligenza Artificiale e Allineamento dei Valori ... 505
15.7.2 Sistemi Autonomi e Controllo [e Fiducia] ... 506
15.7.3 Impatti Economici su Posti di Lavoro, Competenze, Salari ... 507
15.7.4 Comunicazione Scientifica e Alfabetizzazione IA ... 508
15.8 Conclusione ... 509
15.9 Risorse ... 509
16. IA Sostenibile ... 511
16.1 Introduzione ... 511
16.2 Responsabilità Sociale ed Etica ... 512
16.2.1 Considerazioni Etiche ... 512
16.2.2 Sostenibilità a Lungo Termine ... 514
16.2.3 IA per il Bene Ambientale ... 514
16.2.4 Caso di Studio ... 515
16.3 Consumo Energetico ... 515
16.3.1 Comprendere le Esigenze Energetiche ... 515
16.3.2 Data Center e il Loro Impatto ... 517
16.3.3 Ottimizzazione Energetica ... 520
16.4 Impronta di Carbonio ... 520
16.4.1 Definizione e Significato ... 520
16.4.2 La Necessità di Consapevolezza e Azione ... 522
16.4.3 Stima dell'Impronta di Carbonio dell'IA ... 522
16.5 Oltre l'Impronta di Carbonio ... 524
16.5.1 Utilizzo e Stress Idrico ... 525

16.5.2 Uso di Sostanze Chimiche Pericolose 525
16.5.3 Esaurimento delle Risorse 526
16.5.4 Generazione di Rifiuti Pericolosi 526
16.5.5 Impatti sulla Biodiversità 527
16.6 Analisi del Ciclo di Vita 528
16.6.1 Fasi del Ciclo di Vita di un Sistema di IA 528
16.6.2 Impatto Ambientale in Ogni Fase 529
16.7 Sfide nell'LCA 529
16.7.1 Mancanza di Coerenza e Standard 529
16.7.2 Lacune nei Dati 530
16.7.3 Rapido Ritmo di Evoluzione 531
16.7.4 Complessità della Catena di Fornitura 531
16.8 Progettazione e Sviluppo Sostenibili 532
16.8.1 Principi di Sostenibilità 532
16.9 Infrastruttura di IA Green 533
16.9.1 Sistemi di IA a Risparmio Energetico 533
16.9.2 Infrastruttura di IA Sostenibile 534
16.9.3 Framework e Strumenti 535
16.9.4 Benchmark e Classifiche 536
16.10 Caso di Studio: 4M di Google 537
16.10.1 Le 4M Best Practice di Google 537
16.10.2 Risultati Significativi 538
16.10.3 Ulteriori Miglioramenti 538
16.11 IA Embedded - Internet of Trash 539
16.11.1 Rifiuti Elettronici 540
16.11.2 Elettronica Monouso 540
16.11.3 Obsolescenza Programmata 541
16.12 Considerazioni Normative e Politiche 542
16.12.1 Mandati di Misura e Rendicontazione 542
16.12.2 Meccanismi di Restrizione 542
16.12.3 Incentivi Governativi 543
16.12.4 Autoregolamentazione 543
16.12.5 Considerazioni Globali 544
16.13 Percezione e Coinvolgimento del Pubblico 544
16.13.1 Consapevolezza dell'IA 544
16.13.2 Messaggistica 545
16.13.3 Partecipazione Equa 545
16.13.4 Trasparenza 546
16.14 Direzioni e Sfide Future 547
16.14.1 Direzioni Future 547
16.14.2 Sfide 548
16.15 Conclusione 548
16.16 Risorse 549
17. IA Robusta 550
17.1 Introduzione 551
17.2 Esempi del mondo reale 552
17.2.1 Cloud 552
17.2.2 Edge 554
17.2.3 Embedded 555
17.3 Guasti Hardware 556
17.3.1 Guasti Transitori 557
17.3.2 Guasti Permanenti 560
17.3.3 Guasti Intermittenti 563
17.3.4 Rilevamento e Mitigazione 565
17.3.5 Riepilogo 572
17.4 Robustezza del Modello ML 572
17.4.1 Attacchi Avversari 572
17.4.2 Avvelenamento dei Dati 578
17.4.3 Distribution Shift 587
17.4.4 Rilevamento e Mitigazione 592
17.5 Errori Software 599
17.5.1 Definizione e Caratteristiche 599
17.5.2 Meccanismi degli Errori Software nei Framework ML 600
17.5.3 Impatto sui Sistemi ML 601

17.5.4 Rilevamento e Mitigazione ... 602
17.6 Strumenti e Framework ... 604
17.6.1 Modelli di Guasto e Modelli di Errore ... 605
17.6.2 Injection Hardware-based di Guasti ... 606
17.6.3 Strumenti di Injection di Guasti Software-based ... 609
17.6.4 Colmare il Divario tra Modelli di Errore Hardware e Software ... 611
17.7 Conclusione ... 614
17.8 Risorse ... 615
18. IA Generativa ... 616
19. AI for Good ... 617
19.1 Introduzione ... 618
19.2 Agricoltura ... 619
19.3 Assistenza Sanitaria ... 621
19.3.1 Espansione dell'Accesso ... 621
19.3.2 Diagnosi Precoce ... 621
19.3.3 Controllo delle Malattie Infettive ... 622
19.3.4 TinyML Design Contest in Healthcare ... 622
19.4 Scienza ... 623
19.5 Conservazione e Ambiente ... 624
19.6 Risposta ai Disastri ... 624
19.7 Istruzione e Sensibilizzazione ... 625
19.8 Accessibilità ... 626
19.9 Infrastruttura e Pianificazione Urbana ... 626
19.10 Sfide e Considerazioni ... 627
19.11 Conclusione ... 627
19.12 Risorse ... 628
20. Conclusione ... 629
20.1 Introduzione ... 629
20.2 Conoscere l'Importanza dei Dataset ML ... 630
20.3 Esplorare il Panorama dei Framework di IA ... 630
20.4 Comprendere i Fondamenti del Training ML ... 630
20.5 Perseguire l'Efficienza nei Sistemi di IA ... 631
20.6 Ottimizzazione delle Architetture dei Modelli ML ... 631
20.7 Avanzamento dell'Hardware di Elaborazione dell'IA ... 632
20.8 Abbracciare l'Apprendimento "On-Device" ... 633
20.9 Semplificazione delle Operazioni ML ... 633
20.10 Garantire Sicurezza e Privacy ... 633
20.11 Sostenere Considerazioni Etiche ... 634
20.12 Promuovere la Sostenibilità e l'Equità ... 635
20.13 Migliorare la Robustezza e la Resilienza ... 635
20.14 Plasmare il Futuro dei Sistemi ML ... 636
20.15 Applicazione di "AI for Good" ... 636
20.16 Congratulazioni ... 637
21. LABORATORI ... 638
Panoramica ... 639
Obiettivi dell'Apprendimento ... 639
Pubblico Target ... 639
Dispositivi Supportati ... 639
Struttura del Lab ... 640
Sequenza dei Laboratori Consigliata ... 640
Risoluzione dei Problemi e Supporto ... 640
Crediti ... 640
Guida Introduttiva ... 641
Requisiti Hardware ... 641
Requisiti Software ... 642
Connettività di Rete ... 642
Conclusione ... 642
Nicla Vision ... 643
21.1 Prerequisiti ... 643
21.2 Setup ... 643
21.3 Esercizi ... 643

Setup **644**
Introduzione 644
Hardware 645
Due Core Paralleli 645
Memoria 646
Sensori 646
Installazione di Arduino IDE 646
Test del Microfono 647
Test dell'IMU 648
Test del sensore ToF (Time of Flight) [tempo di volo] 649
Test della Fotocamera 651
Installazione di OpenMV IDE 651
Collegamento di Nicla Vision a Edge Impulse Studio 656
Espansione della Nicla Vision Board (opzionale) 658
Conclusione 662
Risorse 662
Classificazione delle Immagini **663**
Introduzione 663
Visione Artificiale 665
Obiettivo del Progetto di Classificazione delle Immagini 665
Raccolta Dati 665
Raccolta del Dataset con OpenMV IDE 666
Addestramento del modello con Edge Impulse Studio 668
Il Dataset 668
Impulse Design 671
Pre-elaborazione delle Immagini 673
Progettazione del Modello 674
Addestramento del Modello 675
Test del Modello 677
Distribuzione del modello 678
Libreria Arduino 678
OpenMV 679
Classificazione delle immagini Benchmark (non ufficiale) 687
Conclusione 688
Risorse 689
Rilevamento degli Oggetti **690**
Introduzione 690
Object Detection e Image Classification 691
Una soluzione innovativa per il rilevamento degli oggetti: FOMO 693
Obiettivo del Progetto di Object Detection 694
Raccolta Dati 694
Raccolta del Dataset con OpenMV IDE 695
Edge Impulse Studio 696
Setup del progetto 696
Caricamento dei dati non etichettati 697
Etichettatura del Dataset 698
Impulse Design 699
Pre-elaborazione di tutti i dataset 700
Progettazione, Addestramento e Test del Modello 701
Modello di test con "Live Classification" 703
Distribuzione del Modello 705
Conclusione 707
Risorse 708
Keyword Spotting (KWS) **709**
Introduzione 709
Come funziona un assistente vocale? 710
Il Progetto Pratico KWS 711
Il Flusso di Lavoro del Machine Learning 711
Il Dataset 711
Caricamento del set di dati su Edge Impulse Studio 712
Acquisizione di Dati Audio Aggiuntivi 713
Creazione di Impulse (Pre-Process / Definizione del Modello) 716

Impulse Design ... 716
Pre-elaborazione (MFCC) ... 716
Andiamo sotto il cofano ... 718
Progettazione e Addestramento del Modello ... 718
Andiamo sotto il cofano ... 720
Test ... 720
Classificazione Live ... 721
Distribuzione e Inferenza ... 721
Post-elaborazione ... 722
Conclusione ... 724
Risorse ... 724
Classificazione del Movimento e Rilevamento delle Anomalie ... 725
Introduzione ... 725
Installazione e test dell'IMU ... 726
Definizione della Frequenza di Campionamento: ... 727
Il Caso di Studio: Trasporto Simulato di Container ... 729
Raccolta Dati ... 729
Collegamento del dispositivo a Edge Impulse ... 730
Raccolta Dati ... 731
Impulse Design ... 734
Panoramica sulla Pre-Elaborazione dei Dati ... 736
Feature Spettrali di EI Studio ... 737
Generazione di feature ... 737
Addestramento dei Modelli ... 739
Test ... 740
Distribuzione ... 740
Inferenza ... 741
Post-elaborazione ... 743
Conclusione ... 743
Applicazioni di Casi ... 743
Custodia per Nicla 3D ... 744
Risorse ... 745
XIAO ESP32S3 ... 746
21.4 Prerequisiti ... 746
21.5 Setup ... 746
21.6 Esercizi ... 746
Setup ... 747
Introduzione ... 747
Installazione di XIAO ESP32S3 Sense su Arduino IDE ... 749
Test della scheda con BLINK ... 750
Collegamento del Modulo Sense (Scheda di Espansione) ... 751
Test del Microfono ... 751
Test della Fotocamera ... 754
Test del WiFi ... 754
Conclusione ... 760
Risorse ... 760
Classificazione delle Immagini ... 761
Introduzione ... 761
Un progetto TinyML di Motion Classification: Frutta contro Verdura ... 762
Addestramento del modello con Edge Impulse Studio ... 763
Raccolta Dati ... 763
Impulse Design ... 764
Training ... 766
Deployment ... 767
Test del modello (inferenza) ... 773
Test con un Modello Più Grande ... 774
Esecuzione dell'inferenza su SenseCraft-Web-Toolkit ... 776
Conclusione ... 779
Risorse ... 779
Rilevamento degli Oggetti ... 780
Introduzione ... 780
Object Detection e Image Classification ... 780

Una soluzione innovativa per il Rilevamento di Oggetti: FOMO 782
Obiettivo del Progetto di Object Detection 782
Raccolta Dati 783
Raccolta di Dataset con XIAO ESP32S3 783
Edge Impulse Studio 784
Setup del progetto 784
Caricamento dei dati non etichettati 785
Etichettatura del Dataset 787
Bilanciamento del dataset e suddivisione Train/Test 788
Impulse Design 789
Pre-elaborazione di tutti i dataset 790
Progettazione, Addestramento e Test del Modello 791
Modello di test con "Live Classification" 793
Deploying del Modello (Arduino IDE) 794
Distribuzione del Modello (SenseCraft-Web-Toolkit) 797
Conclusione 800
Risorse 800
Keyword Spotting (KWS) 801
Introduzione 801
Come funziona un assistente vocale? 802
Il progetto KWS 803
Il Flusso di Lavoro del Machine Learning 803
Il Dataset 804
Acquisizione (offline) di dati audio con XIAO ESP32S3 Sense 805
Salvare campioni audio registrati (dataset) come file audio .wav su una scheda microSD 807
App di Acquisizione dei Dati Audio (offline) 812
Modello di training con Edge Impulse Studio 813
Caricamento dei Dati 813
Creazione di Impulse (Pre-Process / Definizione del Modello) 816
Pre-elaborazione (MFCC) 817
Progettazione e Addestramento del Modello 818
Test 820
Distribuzione e Inferenza 822
Post-elaborazione 825
Conclusione 827
Risorse 827
Classificazione del Movimento e Rilevamento delle Anomalie 828
Introduzione 828
Installazione dell'IMU 829
Il Progetto TinyML Motion Classification 834
Collegamento del dispositivo a Edge Impulse 835
Raccolta Dati 837
Pre-elaborazione dei Dati 839
Progettazione del Modello 840
Impulse Design 840
Generazione di feature 841
Training 842
Test 844
Distribuzione 845
Inferenza 845
Conclusione 848
Risorse 849
Raspberry Pi 850
21.7 Prerequisiti 850
21.8 Setup 850
21.9 Esercizi 850
Setup 851
Introduzione 851
Caratteristiche Principali 852
Modelli Raspberry Pi (trattati in questo libro) 852
Applicazioni di Ingegneria 852
Panoramica Hardware 853

Raspberry Pi Zero 2W ... 853
Raspberry Pi 5 ... 853
Installazione del Sistema Operativo ... 854
Il sistema operativo (SO) ... 854
Installazione ... 855
Configurazione Iniziale ... 856
Accesso Remoto ... 857
Accesso SSH ... 857
Per spegnere il Raspi tramite terminale: ... 858
Trasferire file tra il Raspi e un computer ... 858
Aumentare la memoria SWAP ... 860
Installazione di una Fotocamera ... 861
Installazione di una Webcam USB ... 861
Installazione di un Modulo Fotocamera sulla porta CSI ... 865
Esecuzione del desktop Raspi da remoto ... 867
Aggiornamento e Installazione del Software ... 870
Considerazioni Specifiche del Modello ... 870
Raspberry Pi Zero (Raspi-Zero) ... 870
Raspberry Pi 4 o 5 (Raspi-4 o Raspi-5) ... 870
Classificazione delle Immagini ... 871
Introduzione ... 871
Applicazioni in Scenari del Mondo Reale ... 872
Vantaggi dell'Esecuzione della Classificazione su Dispositivi Edge come Raspberry Pi ... 872
Impostazione dell'Ambiente ... 872
Aggiornamento di Raspberry Pi ... 872
Installazione delle Librerie Richieste ... 873
Impostazione di un Ambiente Virtuale (Facoltativo ma Consigliato) ... 873
Installazione di TensorFlow Lite ... 873
Installazione di Librerie Python Aggiuntive ... 873
Creazione di una directory di lavoro: ... 873
Impostazione di Jupyter Notebook (Facoltativo) ... 874
Verifica della Configurazione ... 875
Fare inferenze con Mobilenet V2 ... 876
Definire una funzione generale di Classificazione delle Immagini ... 879
Test con un modello addestrato da zero ... 880
Installing Picamera2 ... 881
Progetto di Classificazione delle Immagini ... 883
L'Obiettivo ... 883
Raccolta Dati ... 884
Compromesso: Accuratezza contro Velocità ... 890
Test del Modello ... 891
Distribuzione del modello ... 891
Classificazione delle Immagini in Tempo Reale ... 895
Conclusione: ... 900
Risorse ... 901
Rilevamento degli Oggetti ... 902
Introduzione ... 902
Fondamenti dell'Object Detection ... 904
Panoramica dei Modelli di Rilevamento degli Oggetti Pre-Addestrati ... 905
Impostazione dell'Ambiente TFLite ... 906
Creazione di una Directory di Lavoro: ... 906
Inferenza e Post-Elaborazione ... 906
EfficientDet ... 910
Progetto di Rilevamento di Oggetti ... 910
L'Obiettivo ... 910
Raccolta Dati Grezzi ... 911
Etichettatura dei Dati ... 913
Addestramento di un Modello SSD MobileNet su Edge Impulse Studio ... 917
Caricamento dei dati annotati ... 917
Impulse Design ... 918
Pre-elaborazione di tutti i dataset ... 919
Progettazione, Addestramento e Test del Modello ... 920
Distribuzione del modello ... 921
Inferenza e Post-Elaborazione ... 922

Training di un modello FOMO su Edge Impulse Studio 927
Come funziona FOMO? 927
Impulse Design, nuovo Training e Test 928
Distribuzione del modello 930
Inferenza e Post-Elaborazione 931
Esplorazione di un Modello YOLO tramite Ultralitics 933
A proposito del modello YOLO 934
Installazione 936
Test di YOLO 936
Esportazione del Modello in formato NCNN 937
Esplorare YOLO con Python 938
Addestramento di YOLOv8 su un Dataset Personalizzato 940
Inferenza col modello addestrato, usando Raspi 943
Rilevamento di Oggetti su un Live Streaming 944
Conclusione 947
Risorse 948
Small Language Models (SLM) 949
Introduzione 949
Setup 950
Raspberry Pi Active Cooler 951
Generative AI (GenAI) 952
Large Language Models (LLM) 952
Modelli Chiusi e Aperti: 953
Small Language Models (SLM) 953
Ollama 954
Installazione di Ollama 955
Meta Llama 3.2 1B/3B 956
Google Gemma 2 2B 959
Microsoft Phi3.5 3.8B 960
Modelli Multimodali 961
Ispezione delle risorse locali 963
Libreria Python Ollama 964
Chiamata di Funzione 968
1. Importazione delle Librerie 969
2. Definizione di Input e Modello 969
3. Definizione della Struttura dei Dati di Risposta 969
4. Impostazione del Client OpenAI 970
5. Generazione della Risposta 970
6. Calcolo della Distanza 970
Aggiunta di immagini 971
SLM: Tecniche di Ottimizzazione 974
Implementazione del RAG 975
Un semplice progetto RAG 975
Andando Oltre 979
Conclusione 980
Risorse 981
Lab Condivisi 982
KWS Feature Engineering 983
Introduzione 983
Il KWS 984
Introduzione ai Segnali Audio 985
Perché l'Audio Grezzo No? 986
Introduzione agli MFCC 987
Cosa sono gli MFCC? 987
Perché gli MFCC sono importanti? 987
Calcolo degli MFCC 988
Pratica con Python 990
Conclusione 990
Risorse 991
Blocco delle Feature Spettrali DSP 992
Introduzione 992
Estrazione delle Feature di Revisione 993
Un progetto TinyML di Motion Classification 994

Pre-elaborazione dei Dati 995
Edge Impulse - Spectral Analysis Block V.2 funzionamento 996
Feature Statistiche del Dominio del Tempo 999
Feature spettrali 1001
Dominio tempo-frequenza 1003
Wavelet 1003
Analisi Wavelet 1005
Estrazione delle Feature 1006
Conclusione 1008
Riferimenti 1009

Prefazione

Benvenuti in Machine Learning Systems. Questo libro è la porta d'accesso al mondo frenetico dei sistemi di intelligenza artificiale. È un'estensione del corso CS249r alla Harvard University.

Abbiamo creato questo libro open source come sforzo collaborativo per riunire spunti di studenti, professionisti e la più ampia comunità di professionisti dell'IA. Il nostro obiettivo è sviluppare una guida completa che esplori le complessità dei sistemi di intelligenza artificiale e le loro numerose applicazioni.

> "Se vuoi andare veloce, vai da solo. Se vuoi andare lontano, vai insieme ad altre persone". – Proverbio africano

Questo non è un libro statico; è un documento vivo e pulsante. Lo stiamo rendendo open source e lo aggiorniamo costantemente per soddisfare le esigenze in continua evoluzione di questo campo dinamico. Contiene un ricco mix di conoscenze specialistiche che guideranno attraverso la complessa interazione tra algoritmi all'avanguardia e i principi fondamentali che li fanno funzionare. Stiamo preparando il terreno per il prossimo grande balzo nell'innovazione dell'IA.

Perché Abbiamo Scritto Questo Libro

Viviamo in un'epoca in cui la tecnologia è in continua evoluzione. La collaborazione aperta e la condivisione delle conoscenze sono gli elementi costitutivi della vera innovazione. Questo è lo spirito alla base di questo lavoro. Andiamo oltre il tradizionale modello di libro di testo per creare un hub di conoscenza vivo, in modo che possiamo tutti condividere e imparare gli uni dagli altri.

Il libro si concentra sui principi e sui casi di studio dei sistemi di IA, con l'obiettivo di fornire una comprensione approfondita che aiuterà a navigare nel panorama in continua evoluzione dei sistemi di IA. Mantenendolo "open source", non stiamo solo rendendo accessibile l'apprendimento, ma stiamo anche invitando nuove idee e miglioramenti continui. In breve, stiamo costruendo una comunità in cui la conoscenza è libera di crescere e illuminare la strada verso la tecnologia AI globale.

Cosa c’è da Sapere

Per immergersi in questo libro, non si dev’essere un esperto di AI. Tutto ciò di cui c’è bisogno è una conoscenza di base dei concetti di informatica e la curiosità di esplorare su come funzionano i sistemi AI. È qui che avviene l’innovazione e una conoscenza di base della programmazione e delle strutture dati sarà la bussola.

Dichiarazione di Trasparenza dei Contenuti

Questo libro è un progetto guidato dalla comunità, con contenuti generati da numerosi collaboratori nel tempo. Il processo di creazione dei contenuti potrebbe aver coinvolto vari strumenti di editing, tra cui la tecnologia AI generativa. In qualità di autore principale, editore e curatore, il Prof. Vijay Janapa Reddi mantiene la supervisione umana e la supervisione editoriale per garantire che il contenuto sia accurato e pertinente. Tuttavia, nessuno è perfetto, quindi potrebbero comunque esserci delle inesattezze. Apprezziamo molto i feedback e invitiamo a fornire correzioni e suggerimenti. Questo approccio collaborativo è fondamentale per migliorare e mantenere la qualità del contenuto e rendere le informazioni accessibili a livello globale.

Per dare una mano

Se si è interessati a contribuire, le linee guida si trovano qui.

Contatti

Ci sono domande o feedback? Si è liberi di inviare una e-mail al Prof. Vijay Janapa Reddi direttamente, oppure si può avviare un thread di discussione su GitHub.

Collaboratori

Un grande ringraziamento a tutti coloro che hanno contribuito a rendere questo libro quello che è! L'elenco completo dei singoli collaboratori è qui e ulteriori dettagli sullo stile GitHub qui. Benvenuti come collaboratori!

Copyright

Questo libro è open source e sviluppato in modo collaborativo tramite GitHub. Salvo diversa indicazione, questo lavoro è concesso in licenza con Creative Commons Attribuzione-Non commerciale-Condividi allo stesso modo 4.0 Internazionale (CC BY-NC-SA 4.0 CC BY-SA 4.0). Il testo completo della licenza si trova qui.
I collaboratori di questo progetto hanno dedicato i loro contributi al pubblico dominio o con la stessa licenza aperta del progetto originale. Sebbene i contributi siano collaborativi, ogni collaboratore mantiene il copyright sui rispettivi contributi.
Per i dettagli sulla paternità, i contributi e come contribuire, consultare il repository del progetto su GitHub.
Tutti i marchi e i marchi registrati menzionati in questo libro sono di proprietà dei rispettivi proprietari.
Le informazioni fornite in questo libro sono ritenute accurate e affidabili. Tuttavia, gli autori, i curatori e gli editori non possono essere ritenuti responsabili per eventuali danni causati o presumibilmente causati, direttamente o indirettamente, dalle informazioni contenute nel presente libro.

Dedica

Questo libro è una testimonianza dell'idea che, nell'immensità della tecnologia e dell'innovazione, non sono sempre i sistemi più grandi, ma quelli più piccoli, a poter cambiare il mondo.

Ringraziamenti

Mettere insieme questo libro è stato un lungo viaggio, durato diversi anni di duro lavoro. L'idea iniziale per questo libro è nata dal corso TinyML edX e la sua realizzazione non sarebbe stata possibile senza i preziosi contributi di innumerevoli persone. Siamo profondamente in debito con i ricercatori il cui lavoro pionieristico ha gettato le basi per questo libro.

Singoli Collaboratori

Estendiamo la nostra sincera gratitudine alla comunità open source di studenti, insegnanti e contributori. Che abbiate contribuito con un'intera sezione, una singola frase o semplicemente corretto un errore di battitura, i vostri sforzi hanno migliorato questo libro. Apprezziamo profondamente il tempo, la competenza e l'impegno di tutti. Questo libro è tanto vostro quanto nostro.

Un ringraziamento speciale va al professor Vijay Janapa Reddi, la cui convinzione nel potere trasformativo delle comunità open source e la cui guida inestimabile sono state la nostra luce guida fin dall'inizio.

Dobbiamo molto anche al team di GitHub e di Quarto. Avete rivoluzionato il modo in cui le persone collaborano e questo libro è una testimonianza di ciò che si può ottenere quando vengono rimosse le barriere alla cooperazione globale.

Agenzie e Aziende Finanziatrici

Siamo immensamente grati per il generoso supporto delle varie agenzie e aziende finanziatrici che hanno supportato gli assistenti didattici coinvolti in questo lavoro. Le organizzazioni elencate di seguito hanno svolto un ruolo cruciale nel dare vita a questo progetto con i loro contributi.

HDSI | Harvard Data Science Initiative

HARVARD Extension School

Google

NSF

Ai Nostri Lettori

A tutti coloro che acquisteranno questo libro, vogliamo ringraziarvi! L'abbiamo scritto pensando a voi, sperando di provocare riflessioni, ispirare domande e forse anche accendere una scintilla di ispirazione. Dopo tutto, che senso ha scrivere se nessuno legge?

Collaboratori e Ringraziamenti

Estendiamo i nostri più sinceri ringraziamenti al gruppo eterogeneo di persone che hanno generosamente contribuito con la loro competenza, intuizioni, tempo e supporto per migliorare sia il contenuto che la base di codice di questo progetto. Ciò include non solo coloro che hanno contribuito direttamente tramite codice e scrittura, ma anche coloro che hanno aiutato identificando problemi, fornendo feedback e offrendo suggerimenti. Di seguito c'è l'elenco di tutti i collaboratori. Per contribuire a questo progetto, visitare la nostra pagina GitHub per maggiori informazioni.

Vijay Janapa Reddi
Ikechukwu Uchendu
Naeem Khoshnevis
Marcelo Rovai
jasonjabbour
Douwe den Blanken
shanzehbatool
kai4avaya
Elias Nuwara
Jared Ping
Sara Khosravi
Matthew Stewart
Itai Shapira
Maximilian Lam
Jayson Lin
Sophia Cho
Andrea
Jeffrey Ma
Alex Rodriguez
Korneel Van den Berghe
Colby Banbury
Zishen Wan
Abdulrahman Mahmoud
Srivatsan Krishnan
Divya Amirtharaj
Emeka Ezike
Aghyad Deeb
Haoran Qiu
marin-llobet
Emil Njor
Aditi Raju
Jared Ni
Michael Schnebly
oishib
ELSuitorHarvard
Henry Bae
Jae-Won Chung
Yu-Shun Hsiao
Mark Mazumder
Marco Zennaro
Eura Nofshin
Andrew Bass
Pong Trairatvorakul
Jennifer Zhou
Shvetank Prakash
Alex Oesterling
Arya Tschand
Bruno Scaglione
Gauri Jain
Allen-Kuang
Fin Amin
Sercan Aygün
Fatima Shah
gnodipac886
Abenezer
Baldassarre Cesarano
Emmanuel Rassou
Bilge Acun
yanjingl
Yang Zhou
abigailswallow
Jason Yik
happyappledog
Curren Iyer
Jessica Quaye
Sonia Murthy
Shreya Johri
Vijay Edupuganti
The Random DIY
Costin-Andrei Oncescu
Annie Laurie Cook
Jothi Ramaswamy
Batur Arslan
Fatima Shah
a-saraf
songhan
Zishen

Informazioni sul Libro

Panoramica

Benvenuti a questo progetto collaborativo avviato dalla classe CS249r Machine Learning Systems presso l'Università di Harvard. Il nostro obiettivo è rendere questo libro una risorsa della comunità che aiuti educatori e studenti a comprendere i sistemi di ML. Il libro verrà aggiornato regolarmente per riflettere nuove intuizioni sui sistemi ML e metodi di insegnamento efficaci.

Argomenti Esplorati

Questo libro offre una panoramica completa di vari aspetti dei sistemi di apprendimento automatico. Copriamo l'intero flusso di lavoro dei sistemi ML end-to-end, iniziando con concetti fondamentali e procedendo attraverso l'ingegneria dei dati, i framework AI e il training dei modelli.
Si imparerà ad ottimizzare i modelli per l'efficienza, a distribuire l'AI su varie piattaforme hardware e a confrontare le prestazioni. Il libro esplora anche argomenti più avanzati come la sicurezza, la privacy, l'intelligenza artificiale responsabile e sostenibile, l'IA solida e generativa e l'impatto sociale dell'IA. Alla fine, si avranno solide basi e approfondimenti pratici sia sulle dimensioni tecniche che etiche dell'apprendimento automatico.
Ci auguriamo che una volta terminato questo libro si abbia una conoscenza di base dell'apprendimento automatico e delle sue applicazioni. Si impareranno anche le implementazioni reali dei sistemi di apprendimento automatico e si acquisirà esperienza pratica tramite laboratori e compiti basati su progetti.

Chi Dovrebbe Leggerlo

Questo libro è pensato per le persone in varie fasi della loro interazione con i sistemi di apprendimento automatico. Inizia con i fondamenti e passa ad argomenti più avanzati pertinenti alla comunità ML e ad aree di ricerca più ampie. I destinatari più rilevanti includono:

- **Studenti di Informatica e Ingegneria Elettrica:** Studenti laureati e laureati in questi campi troveranno questo libro inestimabile. Presenta le tecniche utilizzate nella progettazione e nella creazione di sistemi ML, concentrandosi sui fondamenti piuttosto che sulla profondità, in genere il focus dell'istruzione in aula. Questo libro mira a fornire il background e il contesto necessari, consentendo agli insegnanti di approfondire argomenti avanzati. Un aspetto importante è l'attenzione end-to-end, spesso trascurata nei programmi tradizionali.
- **Ingegneri di Sistema:** Per gli ingegneri, questo libro funge da guida per comprendere le sfide delle applicazioni intelligenti, in particolare su piattaforme ML con risorse limitate. Copre il framework concettuale e i componenti pratici che costituiscono un sistema ML, estendendosi oltre le aree specifiche in cui ci si potrebbe specializzare nel proprio lavoro.
- **Ricercatori e Accademici:** I ricercatori scopriranno che questo libro affronta le sfide uniche dell'esecuzione di algoritmi di apprendimento automatico su diverse piattaforme. L'efficienza sta diventando sempre più importante; la sola comprensione degli algoritmi non è sufficiente, poiché una comprensione più approfondita dei sistemi è necessaria per costruire modelli più efficienti. Per i ricercatori, il libro cita articoli seminali, guidando verso lavori fondamentali che hanno plasmato il campo e tracciando connessioni tra varie aree con implicazioni significative per il proprio lavoro.

Principali Risultati dell'Apprendimento

I lettori acquisiranno competenze nel training e nell'implementazione di modelli di reti neurali profonde su diverse piattaforme, oltre a comprendere le sfide più ampie coinvolte nella loro progettazione, sviluppo e implementazione. Nello specifico, dopo aver completato questo libro, gli studenti saranno in grado di:

1. Spiegare i concetti fondamentali e la loro rilevanza per i sistemi di IA.
2. Descrivere i componenti fondamentali e l'architettura dei sistemi di IA.
3. Confrontare e mettere a contrasto varie piattaforme hardware per l'implementazione dell'IA, selezionando le opzioni appropriate per casi d'uso specifici.
4. Progettare e implementare processi di training per modelli di IA su sistemi diversi.
5. Applicare tecniche di ottimizzazione per migliorare le prestazioni e l'efficienza dei modelli di IA.
6. Analizzare le applicazioni di IA nel mondo reale e le loro strategie di implementazione.
7. Valutare le attuali sfide nei sistemi di IA e prevedere le tendenze future nel settore.
8. Sviluppare un progetto completo basato sull'apprendimento automatico, dalla concezione all'implementazione.
9. Risolvere i problemi comuni nell'addestramento e nell'implementazione dei modelli di IA.
10. Valutare criticamente le implicazioni etiche e gli impatti sociali dei sistemi di IA.

Prerequisiti per i Lettori

- **Competenze di programmazione di base:** Consigliamo di avere una certa esperienza di programmazione, idealmente in Python. Una conoscenza delle variabili, tipi di dati e strutture di controllo faciliterà l'interazione col libro.
- **Alcune Conoscenze di Machine Learning:** Sebbene non sia obbligatorio, una conoscenza di base dei concetti di apprendimento automatico aiuterà ad assorbire il materiale più facilmente. Se si è nuovi nel campo, il libro fornisce sufficienti informazioni di base per mettersi al passo.
- **Conoscenza di Base dei Sistemi:** Si consiglia un livello di conoscenza di base dei sistemi a livello universitario junior o senior. Sarà utile comprendere l'architettura di sistema, i sistemi operativi e le reti di base.
- **Programmazione Python (Facoltativo):** Se si ha familiarità con Python, si troverà più facile interagire con le sezioni di codifica del libro. Conoscere librerie come NumPy, scikit-learn e TensorFlow sarà particolarmente utile.
- **Voglia di Imparare:** Il libro è progettato per essere accessibile a un vasto pubblico, con diversi livelli di competenza tecnica. La volontà di sfidare se stessi e di impegnarsi in esercizi pratici aiuterà a trarne il massimo vantaggio.
- **Disponibilità delle Risorse:** Per gli aspetti pratici, ci sarà bisogno di un computer con Python e le librerie pertinenti installate. L'accesso facoltativo a schede di sviluppo o hardware specifico sarà utile anche per sperimentare la distribuzione del modello di apprendimento automatico.

Soddisfacendo questi prerequisiti, si sarà ben posizionati per approfondire la comprensione dei sistemi di apprendimento automatico, impegnarsi in esercizi di codifica e persino implementare applicazioni pratiche su vari dispositivi.

1. Introduzione

DALL·E 3 Prompt: Un'illustrazione 2D dettagliata, rettangolare e piatta che raffigura una roadmap dei capitoli di un libro sui sistemi di apprendimento automatico, su uno sfondo bianco nitido e pulito. L'immagine presenta una strada tortuosa che attraversa vari punti di riferimento simbolici. Ogni punto di riferimento rappresenta un argomento del capitolo: Introduzione, Sistemi di ML, Avvio al Deep Learning, Workflow dell'IA, Data Engineering, Framework di IA, Addestramento dell'IA, IA Efficiente, Ottimizzazioni dei Modelli, Accelerazione IA, Benchmarking dell'IA, Apprendimento On-Device, Operazioni di ML, Sicurezza e Privacy, IA Responsabile, IA Sostenibile, AI for Good, IA Robusta, IA Generativa. Lo stile è pulito, moderno e piatto, adatto a un libro tecnico, con ogni punto di riferimento chiaramente etichettato con il titolo del capitolo.

1.1 Panoramica

All'inizio degli anni '90, Mark Weiser, un pioniere dell'informatica, ha introdotto il mondo a un concetto rivoluzionario che avrebbe cambiato per sempre il modo in cui interagiamo con la tecnologia. Questa visione è stata riassunta in modo sintetico nel suo articolo fondamentale, "The Computer for the 21st Century" (Figura 1.1). Weiser immaginava un futuro in cui l'informatica sarebbe stata perfettamente integrata nei nostri ambienti, diventando una parte invisibile e integrante della vita quotidiana.

The Computer for the 21st Century

Specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence

by Mark Weiser

The most profound technologies are those that disappear. They weave themselves into the fabric of everyday life until they are indistinguishable from it.

Consider writing, perhaps the first information technology. The ability to represent spoken language symbolically for long-term storage freed information from the limits of individual memory. Today this technology is ubiquitous in industrialized countries. Not only do books, magazines and newspapers convey written information, but so do street signs, billboards, shop signs and even graffiti. Candy wrappers are covered in writing. The constant background presence of these products of "literacy technology" does not require active attention, but the information to be transmitted is ready for use at a glance. It is difficult to imagine modern life otherwise.

Silicon-based information technology, in contrast, is far from having become part of the environment. More than 50 million personal computers have been sold, and the computer nonetheless remains largely in a world of its own. It is approachable only through complex jargon that has nothing to do with the tasks for which people use computers. The state of the art is perhaps analogous to the period when scribes had to know as much about making ink or baking clay as they did about writing.

The arcane aura that surrounds personal computers is not just a "user interface" problem. My colleagues and I at the Xerox Palo Alto Research Center think that the idea of a "personal" computer itself is misplaced and that the vision of laptop machines, dynabooks and "knowledge navigators" is only a transitional step toward achieving the real potential of information technology. Such machines cannot truly make computing an integral, invisible part of people's lives. We are therefore trying to conceive a new way of thinking about computers, one that takes into account the human world and allows the computers themselves to vanish into the background.

Such a disappearance is a fundamental consequence not of technology but of human psychology. Whenever people learn something sufficiently well, they cease to be aware of it. When you look at a street sign, for example, you absorb its information without consciously performing the act of reading. Computer scientist, economist and Nobelist Herbert A. Simon calls this phenomenon "compiling"; philosopher Michael Polanyi calls it the "tacit dimension"; psychologist J. J. Gibson calls it "visual invariants"; philosophers Hans Georg Gadamer and Martin Heidegger call it the "horizon" and the "ready-to-hand"; John Seely Brown of PARC calls it the "periphery." All say, in essence, that only when things disappear in this way are we freed to use them without thinking and so to focus beyond them on new goals.

The idea of integrating computers seamlessly into the world at large runs counter to a number of present-day trends. "Ubiquitous computing" in this context does not mean just computers that can be carried to the beach, jungle or airport. Even the most powerful notebook computer, with access to a worldwide information network, still focuses attention on a single box. By analogy with writing, carrying a superlaptop is like owning just one very important book. Customizing this book, even writing millions of other books, does not begin to capture the real power of literacy.

Furthermore, although ubiquitous computers may use sound and video in addition to text and graphics, that does not make them "multimedia computers." Today's multimedia machine makes the computer screen into a demanding focus of attention rather than allowing it to fade into the background.

Perhaps most diametrically opposed to our vision is the notion of virtual reality, which attempts to make a world inside the computer. Users don special goggles that project an artificial scene onto their eyes; they wear gloves or even bodysuits that sense their motions and gestures so that they can move about and manipulate virtual objects. Although it may have its purpose in allowing people to explore realms otherwise inaccessible—the insides of cells, the surfaces of distant planets, the information web of data bases—virtual reality is only a map, not a territory. It excludes desks, offices, other people not wearing goggles and bodysuits, weather, trees, walks, chance encounters and, in general, the infinite richness of the universe. Virtual reality focuses an enormous apparatus on simulating the world rather than on invisibly enhancing the world that already exists.

Indeed, the opposition between the

MARK WEISER is head of the Computer Science Laboratory at the Xerox Palo Alto Research Center. He is working on the next revolution of computing after workstations, variously known as ubiquitous computing or embodied virtuality. Before working at PARC, he was a professor of computer science at the University of Maryland; he received his Ph.D. from the University of Michigan in 1979. Weiser also helped found an electronic publishing company and a video arts company and claims to enjoy computer programming "for the fun of it." His most recent technical work involved the implementation of new theories of automatic computer memory reclamation, known in the field as garbage collection.

94 SCIENTIFIC AMERICAN *September 1991*

Figura 1.1: Ubiquitous computing as envisioned di Mark Weiser.

Ha definito questo concetto "ubiquitous computing", promettendo un mondo in cui la tecnologia ci avrebbe servito senza richiedere la nostra costante attenzione o interazione.
Facciamo un salto in avanti fino a oggi, e ci troviamo sul punto di realizzare la visione di Weiser, grazie all'avvento e alla proliferazione dei sistemi di machine learning [apprendimento automatico].
Nella visione dell'ubiquitous computing (Weiser 1991), l'integrazione dei processori negli oggetti di uso quotidiano è solo un aspetto di un più ampio cambiamento di paradigma. La vera essenza di questa visione risiede nella creazione di un ambiente intelligente in grado di anticipare le nostre esigenze e agire per nostro conto, migliorando le nostre esperienze senza richiedere comandi espliciti. Per raggiungere questo livello di intelligenza pervasiva, è fondamentale sviluppare e distribuire sistemi di machine learning che coprano l'intero ecosistema, dal cloud all'edge e persino ai più piccoli dispositivi IoT.
Distribuendo le capacità di apprendimento automatico lungo tutto il "continuum informatico", dal cloud all'edge fino ai sistemi embedded che ci circondano, possiamo sfruttare i punti di forza di ogni layer mitigandone al contempo i limiti. Il cloud, con le sue vaste risorse di elaborazione e capacità di archiviazione, è ideale per addestrare modelli complessi su grandi set di dati ed eseguire attività che richiedono molte risorse. I dispositivi edge, come gateway e smartphone, possono elaborare i dati localmente, consentendo tempi di risposta più rapidi, una migliore privacy e requisiti di larghezza di banda ridotti. Infine, i dispositivi IoT più piccoli, dotati di capacità di apprendimento automatico, possono prendere decisioni rapide in base ai dati dei sensori, consentendo sistemi altamente reattivi ed efficienti.
Questa intelligenza distribuita è particolarmente cruciale per le applicazioni che richiedono elaborazione in tempo reale, come veicoli autonomi, automazione industriale e assistenza

sanitaria smart [intelligente]. Elaborando i dati al livello più appropriato del continuum informatico, possiamo garantire che le decisioni vengano prese in modo rapido e accurato, senza fare affidamento su una comunicazione costante con un server centrale.
La migrazione dell'intelligenza di apprendimento automatico nell'ecosistema consente inoltre esperienze più personalizzate e consapevoli del contesto. Imparando dal comportamento e dalle preferenze degli utenti all'edge, i dispositivi possono adattarsi alle esigenze individuali senza compromettere la privacy. Questa intelligenza localizzata può quindi essere aggregata e perfezionata nel cloud, creando un ciclo di feedback che migliora costantemente il sistema complessivo.
Tuttavia, l'implementazione di sistemi di apprendimento automatico nel continuum informatico presenta diverse sfide. Garantire l'interoperabilità e l'integrazione senza soluzione di continuità di questi sistemi richiede protocolli e interfacce standardizzati. È inoltre necessario affrontare i problemi di sicurezza e privacy, poiché la distribuzione dell'intelligenza su più livelli aumenta la superficie di attacco e il potenziale di violazioni dei dati.
Inoltre, le diverse capacità computazionali e i vincoli energetici dei dispositivi a diversi livelli del continuum informatico richiedono lo sviluppo di modelli di apprendimento automatico efficienti e adattabili. Tecniche come la compressione del modello, il "federated learning" [apprendimento federato] e il transfer learning [apprendimento tramite trasferimento] possono aiutare ad affrontare queste sfide, consentendo l'implementazione dell'intelligenza su un'ampia gamma di dispositivi.
Mentre ci avviciniamo alla realizzazione della visione di Weiser dell'ubiquitous computing, lo sviluppo e l'implementazione di sistemi di apprendimento automatico nell'intero ecosistema saranno fondamentali. Sfruttando i punti di forza di ogni layer [livello] del continuum informatico, possiamo creare un ambiente intelligente che si integra perfettamente con la nostra vita quotidiana, anticipando le nostre esigenze e migliorando le nostre esperienze in modi che un tempo erano inimmaginabili. Mentre continuiamo a spingere i confini di ciò che è possibile con l'apprendimento automatico distribuito, ci avviciniamo sempre di più a un futuro in cui la tecnologia diventa una parte invisibile ma integrante del nostro mondo.

Figura 1.2: Applicazioni comuni di Machine Learning. Fonte: EDUCBA

Questa visione sta già iniziando a prendere forma, come illustrato dalle comuni applicazioni dell'IA che ci circondano nella nostra vita quotidiana (Figura 1.2). Dall'assistenza sanitaria e

dalla finanza ai trasporti e all'intrattenimento, l'apprendimento automatico sta trasformando vari settori, rendendo le nostre interazioni con la tecnologia più intuitive e personalizzate.

1.2 Cosa c'è nel Libro

In questo libro, esploreremo le basi tecniche dei sistemi di apprendimento automatico onnipresenti, le sfide della creazione e distribuzione di questi sistemi nel continuum informatico e la vasta gamma di applicazioni che consentono. Un aspetto unico di questo libro è la sua funzione di canale verso opere accademiche fondamentali e documenti di ricerca accademica, mirati ad arricchire la comprensione del lettore e incoraggiare un'esplorazione più approfondita dell'argomento. Questo approccio cerca di colmare il divario tra materiali pedagogici e tendenze di ricerca all'avanguardia, offrendo una guida completa che è al passo con l'evoluzione del campo dell'apprendimento automatico applicato.

Per migliorare l'esperienza di apprendimento, abbiamo incluso una varietà di materiali supplementari. In tutto il libro, si troveranno slide che riassumono i concetti chiave, video che forniscono spiegazioni e dimostrazioni approfondite, esercizi che rafforzano la comprensione ed esercizi pratici che offrono esperienza pratica con gli strumenti e le tecniche discussi. Queste risorse aggiuntive sono progettate per soddisfare diversi stili di apprendimento e contribuire ad acquisire una comprensione più profonda e pratica dell'argomento.

Iniziamo con i fondamenti, introducendo concetti chiave nei sistemi e nell'apprendimento automatico e fornendo un avvio al deep learning. Poi guidiamo attraverso il flusso di lavoro dell'IA, dall'ingegneria dei dati alla selezione dei framework di IA giusti. Questo flusso di lavoro segue da vicino il ciclo di vita di un tipico progetto di apprendimento automatico, come illustrato in Figura 1.3.

Figura 1.3: Ciclo di vita del progetto di apprendimento automatico. Fonte:Medium

La sezione sul training copre le diverse tecniche di training dell'IA efficienti, ottimizzazioni dei modelli e accelerazione dell'IA tramite hardware specializzato. Successivamente si affronta il deployment distribuzione, con capitoli sul benchmarking dell'IA, l'apprendimento distribuito e le operazioni di ML. Argomenti avanzati come sicurezza, privacy, IA responsabile, IA sostenibile, IA robusta e IA generativa vengono quindi esplorati in profondità. Il libro si conclude evidenziando l'impatto positivo dell'IA e il suo potenziale per il bene.

1.3 Come Orientarsi in Questo Libro

Per ottenere il massimo da questo libro, consigliamo un approccio di apprendimento strutturato che sfrutti le varie risorse fornite. Ogni capitolo include slide, video, esercizi e laboratori per soddisfare diversi stili di apprendimento e rafforzare la comprensione. Inoltre, un bot tutor AI (SocratiQ AI) è prontamente disponibile per guidare attraverso i contenuti e fornire assistenza personalizzata.

1. **I Fondamenti (Capitoli 1-3):** Si inizia costruendo una solida base con i primi capitoli, che forniscono un'introduzione all'IA embedded e trattano argomenti fondamentali come sistemi di IA e deep learning.
2. **Flusso di Lavoro (Capitoli 4-6):** Con questa base, si passa ai capitoli incentrati sugli aspetti pratici del processo di creazione del modello AI come flussi di lavoro, ingegneria dei dati e framework.
3. **Training (Capitoli 7-10):** Questi capitoli offrono approfondimenti su come addestrare efficacemente i modelli AI, comprese tecniche per efficienza, ottimizzazioni e accelerazione.

4. **Deployment (Capitoli 11-13):** Si esamina come distribuire l'IA sui dispositivi e monitorarne l'operatività tramite metodi come benchmarking, on-device learning e MLOps.
5. **Argomenti Avanzati (Capitoli 14-18):** Si esaminano criticamente argomenti come sicurezza, privacy, etica, sostenibilità, robustezza e IA generativa.
6. **Impatto Sociale (Capitolo 19):** Esplora le applicazioni positive e il potenziale dell'IA per il bene della società.
7. **Conclusione (Capitolo 20):** Riflessioni sui principali risultati e sulle direzioni future dei sistemi di IA.

Sebbene il libro sia progettato per un apprendimento progressivo, incoraggiamo un approccio di apprendimento interconnesso che consente di navigare tra i capitoli in base ai propri interessi e alle proprie esigenze. In tutto il libro si trovano casi di studio ed esercizi pratici che aiuteranno a mettere in relazione la teoria con le applicazioni del mondo reale. Consigliamo inoltre di partecipare a forum e gruppi per partecipare a discussioni, discutere concetti e condividere approfondimenti con altri studenti. Rivedere regolarmente i capitoli può aiutare a rafforzare l'apprendimento e offrire nuove prospettive sui concetti trattati. Adottando questo approccio strutturato ma flessibile e interagendo attivamente con i contenuti e la community, si farà un'esperienza di apprendimento appagante e arricchente che massimizza la comprensione.

1.4 Suddivisione dei Capitoli

Ecco uno sguardo più da vicino a cosa tratta ogni capitolo. Abbiamo strutturato il libro in sei sezioni principali: Nozioni Fondamentali, Flusso di lavoro, Training, Deployment, Argomenti avanzati e Impatto. Queste sezioni riflettono da vicino i componenti principali di una tipica pipeline di machine learning, dalla comprensione dei concetti di base al la deploy e alla manutenzione dei sistemi di intelligenza artificiale in applicazioni del mondo reale. Organizzando il contenuto in questo modo, puntiamo a fornire una progressione logica che rispecchi il processo effettivo di sviluppo e implementazione dei sistemi di intelligenza artificiale.

1.4.1 Nozioni Fondamentali

Nella sezione Nozioni Fondamentali, poniamo le basi per comprendere l'intelligenza artificiale. Questo è ben lungi dall'essere un'immersione profonda negli algoritmi, ma puntiamo a introdurre concetti chiave, fornire una panoramica dei sistemi di apprendimento automatico e approfondire i principi e gli algoritmi di deep learning che alimentano le applicazioni di IA nei loro sistemi associati. Questa sezione fornisce le conoscenze essenziali necessarie per comprendere i capitoli successivi.

1. **Introduzione:** Questo capitolo prepara il terreno, fornendo una panoramica dell'intelligenza artificiale e gettando le basi per i capitoli successivi.
2. **Sistemi di ML:** Introduciamo le basi dei sistemi di machine learning [apprendimento automatico], le piattaforme in cui gli algoritmi di intelligenza artificiale sono ampiamente applicati.
3. **Avvio al Deep Learning:** Questo capitolo offre una breve introduzione agli algoritmi e ai principi alla base delle applicazioni di IA nei sistemi di apprendimento automatico.

1.4.2 Workflow

La sezione Workflow [Flusso di lavoro] guida attraverso gli aspetti pratici della creazione di modelli AI. Analizziamo il flusso di lavoro AI, discutiamo le "best practice" di data engineering e passiamo in rassegna i framework AI più diffusi. Alla fine di questa sezione, si avrà una

chiara comprensione dei passaggi coinvolti nello sviluppo di applicazioni AI competenti e degli strumenti disponibili per semplificare il processo.

4. **Workflow IA:** Questo capitolo analizza il flusso di lavoro di apprendimento automatico, offrendo approfondimenti sui passaggi che portano ad applicazioni AI competenti.
5. **Ingegneria dei Dati:** Ci concentriamo sull'importanza dei dati nei sistemi di IA, discutendo su come gestire e organizzare efficacemente i dati.
6. **Framework di IA:** Questo capitolo esamina diversi framework per lo sviluppo di modelli di apprendimento automatico, guidando nella scelta di quello più adatto ai propri progetti.

1.4.3 Training

Nella sezione Training, esploriamo tecniche per il training efficiente e affidabile di modelli di IA. Trattiamo strategie per raggiungere efficienza, ottimizzazioni dei modelli e il ruolo dell'hardware specializzato nell'accelerazione IA. Questa sezione fornisce le conoscenze necessarie per sviluppare modelli ad alte prestazioni che possono essere integrati perfettamente nei sistemi di IA.

7. **Training IA:** Questo capitolo approfondisce il training [addestramento] dei modelli, esplorando tecniche per sviluppare modelli efficienti e affidabili.
8. **IA Efficiente:** Qui, discutiamo strategie per raggiungere l'efficienza nelle applicazioni di IA, dall'ottimizzazione delle risorse computazionali al miglioramento delle prestazioni.
9. **Ottimizzazioni dei Modelli:** Esploriamo vari percorsi per ottimizzare i modelli di IA per un'integrazione senza soluzione di continuità nei sistemi di IA.
10. **Accelerazione dell'IA:** Discutiamo il ruolo dell'hardware specializzato nel migliorare le prestazioni dei sistemi di IA.

1.4.4 Deployment

La sezione Deployment distribuzione si concentra sulle sfide e sulle soluzioni per l'implementazione di modelli di IA. Discutiamo metodi di benchmarking per valutare le prestazioni del sistema AI, tecniche per l'apprendimento "on-device" per migliorare l'efficienza e la privacy e i processi coinvolti nelle operazioni di ML. Questa sezione fornisce le competenze per implementare e gestire in modo efficace le funzionalità di IA nei sistemi di intelligenza artificiale.

11. **Benchmark dell'IA:** Questo capitolo si concentra su come valutare i sistemi di IA tramite metodi di benchmarking sistematici.
12. **Apprendimento On-Device:** Esploriamo tecniche per l'apprendimento localizzato, che migliora sia l'efficienza che la privacy.
13. **Operazioni di ML:** Questo capitolo esamina i processi coinvolti nell'integrazione, nel monitoraggio e nella manutenzione senza soluzione di continuità delle funzionalità di IA.

1.4.5 Argomenti Avanzati

Nella sezione Argomenti avanzati, studieremo le problematiche critiche che circondano l'IA. Affrontiamo le preoccupazioni relative a privacy e sicurezza, esploriamo i principi etici dell'intelligenza artificiale responsabile, discutiamo strategie per uno sviluppo sostenibile dell'intelligenza artificiale, esaminiamo tecniche per la creazione di modelli di intelligenza artificiale solidi e introduciamo l'entusiasmante campo dell'intelligenza artificiale generativa. Questa sezione amplia la comprensione del complesso panorama dell'IA e prepara ad affrontarne le sfide.

14. **Sicurezza e Privacy:** Man mano che l'intelligenza artificiale diventa sempre più onnipresente, questo capitolo affronta gli aspetti cruciali della privacy e della sicurezza nei sistemi di IA.
15. **IA Responsabile:** Discutiamo i principi etici che guidano l'uso responsabile dell'intelligenza artificiale, concentrandoci sulla correttezza, responsabilità e trasparenza.
16. **IA Sostenibile:** Questo capitolo esplora pratiche e strategie per un'intelligenza artificiale sostenibile, garantendo fattibilità a lungo termine e un impatto ambientale ridotto.
17. **IA Robusta:** Parliamo di tecniche per sviluppare modelli di IA affidabili e robusti che possano funzionare in modo coerente in varie condizioni.
18. **IA Generativa:** Questo capitolo esplora gli algoritmi e le tecniche alla base dell'IA generativa, aprendo strade all'innovazione e alla creatività.

1.4.6 Impatto Sociale

La sezione Impatto Sociale evidenzia il potenziale trasformativo dell'IA in vari domini. Presentiamo applicazioni reali di TinyML in sanità, agricoltura, conservazione e altre aree in cui l'IA sta facendo una positiva differenza. Questa sezione invoglia a sfruttare la potenza dell'AI per il bene della società e a contribuire allo sviluppo di soluzioni di impatto.

19. **AI for Good:** Evidenziamo applicazioni positive di TinyML in aree come sanità, agricoltura e la conservazione.

1.4.7 Chiusura

Nella sezione Chiusura, riflettiamo sugli insegnamenti chiave del libro e guardiamo al futuro dell'IA. Sintetizziamo i concetti trattati, discutiamo le tendenze emergenti e forniamo indicazioni su come proseguire nel percorso di apprendimento in questo campo in rapida evoluzione. Questa sezione lascia con una comprensione completa dell'IA e l'entusiasmo di applicare le conoscenze in modi innovativi.

20. **Conclusione:** Il libro si conclude con una riflessione sugli apprendimenti chiave e sulle direzioni future nel campo dell'IA.

1.5 Contributi dei Lettori

L'apprendimento nel mondo frenetico dell'intelligenza artificiale è un viaggio collaborativo. Ci siamo prefissati di coltivare una vivace comunità di studenti, innovatori e collaboratori. Esplorando i concetti e impegnandosi con gli esercizi, incoraggiamo a condividere le intuizioni ed esperienze personali. Che si tratti di un approccio innovativo, di un'applicazione interessante o di una domanda stimolante, i contributi dei singoli possono arricchire l'ecosistema di apprendimento. Partecipare alle discussioni, offrire e cercare indicazioni e collaborare a progetti per promuovere una cultura di crescita e apprendimento reciproci. Condividendo la conoscenza, si svolge un ruolo importante nel promuovere una comunità connessa, informata e potenziata a livello globale.

2. Sistemi di ML

DALL·E 3 Prompt: Illustrazione in formato rettangolare che raffigura la fusione di sistemi embedded con IA Embedded. La metà sinistra dell'immagine ritrae i tradizionali sistemi embedded, inclusi microcontrollori e processori, in modo dettagliato e preciso. La metà destra mostra il mondo dell'intelligenza artificiale, con rappresentazioni astratte di modelli di machine learning, neuroni e flusso di dati. Le due metà sono nettamente separate, sottolineando il significato individuale della tecnologia embedded e dell'IA, ma si uniscono in armonia al centro.

I sistemi di Machine Learning(ML), costruiti sulle fondamenta dei sistemi di elaborazione, hanno il potenziale per trasformare il nostro mondo. Questi sistemi, con i loro ruoli specializzati e le capacità di elaborazione in tempo reale, rappresentano un punto di incontro critico in cui dati e elaborazione si incontrano su scala micro. Sono specificamente progettati per ottimizzare le prestazioni, l'uso di energia e l'efficienza spaziale, fattori chiave essenziali per l'implementazione di successo dei sistemi di ML.
Nel corso di questo capitolo esploreremo il mondo complesso e affascinante dei sistemi di apprendimento automatico. Acquisiremo informazioni sulla loro progettazione strutturale e sulle caratteristiche operative e comprenderemo il loro ruolo chiave nel potenziare le applicazioni ML. A partire dalle basi delle unità microcontrollore, esamineremo le interfacce e le periferiche che ne migliorano le funzionalità. Questo capitolo è concepito come una guida completa che spiega gli aspetti più sfumati dei diversi sistemi di apprendimento automatico.

Obiettivi dell'Apprendimento

- Comprendere le caratteristiche e le differenze chiave tra i sistemi Cloud ML, Edge ML e TinyML.
- Analizzare i vantaggi e le sfide associati a ciascun paradigma ML.
- Esplorare applicazioni e casi d'uso reali per Cloud ML, Edge ML e TinyML.
- Confrontare gli aspetti prestazionali di ciascun approccio ML, tra cui latenza, privacy e utilizzo delle risorse.
- Esaminare il panorama in evoluzione dei sistemi ML e i potenziali sviluppi futuri.

2.1 Introduzione

Il ML si sta evolvendo rapidamente, con nuovi paradigmi che plasmano il modo in cui i modelli vengono sviluppati, addestrati e implementati. Il settore sta vivendo un'innovazione significativa guidata dai progressi in hardware, software e tecniche algoritmiche. Questi sviluppi stanno consentendo l'applicazione dell'apprendimento automatico in diversi contesti, dalle infrastrutture cloud su larga scala ai dispositivi edge e persino ad ambienti minuscoli e con risorse limitate.

I moderni sistemi di apprendimento automatico abbracciano uno spettro di opzioni di distribuzione, ciascuna con il proprio set di caratteristiche e casi d'uso. Da un lato, abbiamo l'apprendimento automatico basato su cloud, che sfrutta potenti risorse di elaborazione centralizzate per attività complesse e ad alta intensità di dati. Muovendoci lungo lo spettro, incontriamo l'apprendimento automatico edge, che avvicina l'elaborazione alla fonte dei dati per una latenza ridotta e una migliore privacy. All'estremità opposta, troviamo TinyML, che consente l'apprendimento automatico su dispositivi a bassissimo consumo energetico con pesanti vincoli di memoria ed elaborazione.

Questo capitolo esplora il panorama dei sistemi di apprendimento automatico contemporanei, coprendo tre approcci chiave: Cloud ML, Edge ML e TinyML. Figura 2.1 illustra lo spettro dell'intelligenza distribuita attraverso questi approcci, fornendo un confronto visivo delle loro caratteristiche. Esamineremo le caratteristiche uniche, i vantaggi e le sfide di ogni approccio, come illustrato nella figura. Inoltre, discuteremo le tendenze e le tecnologie emergenti che stanno plasmando il futuro dell'implementazione dell'apprendimento automatico, considerando come potrebbero influenzare l'equilibrio tra questi tre paradigmi.

Figura 2.1: Cloud vs. Edge vs. TinyML: Lo Spettro dell'Intelligenza Distribuita. Fonte: ABI Research – TinyML.

L'evoluzione dei sistemi di apprendimento automatico può essere vista come una progressione dai paradigmi di elaborazione centralizzati a quelli distribuiti:

1. **Cloud ML:** Inizialmente, il ML era prevalentemente basato sul cloud. Per addestrare ed eseguire grandi modelli ML venivano utilizzati server potenti nei data center. Questo approccio sfrutta vaste risorse di elaborazione e capacità di archiviazione, consentendo lo sviluppo di modelli complessi addestrati su enormi set di dati. Cloud ML eccelle nelle attività che richiedono un'ampia potenza di elaborazione ed è ideale per applicazioni in cui la reattività in tempo reale non è critica.

2. **Edge ML:** Con l'aumento della necessità di elaborazione in tempo reale e a bassa latenza, è emerso Edge ML. Questo paradigma avvicina le capacità di inferenza alla fonte dei dati, in genere su dispositivi edge come smartphone, telecamere intelligenti o gateway IoT. Edge ML riduce la latenza, migliora la privacy mantenendo i dati locali e può funzionare con connettività cloud intermittente. È particolarmente utile per applicazioni che richiedono risposte rapide o che gestiscono dati sensibili.
3. **TinyML:** L'ultimo sviluppo in questa direzione è TinyML, che consente l'esecuzione di modelli ML su microcontrollori con risorse estremamente limitate e piccoli sistemi embedded. TinyML consente l'inferenza sul dispositivo senza fare affidamento sulla connettività al cloud o all'edge, aprendo nuove possibilità per dispositivi intelligenti alimentati a batteria. Questo approccio è fondamentale per le applicazioni in cui dimensioni, consumo energetico e costi sono fattori critici.

Ognuno di questi paradigmi ha i suoi punti di forza ed è adatto a diversi casi d'uso:

- Cloud ML rimane essenziale per le attività che richiedono un'enorme potenza di calcolo o analisi di dati su larga scala.
- Edge ML è ideale per le applicazioni che necessitano di risposte a bassa latenza o elaborazione di dati locali.
- TinyML abilita le funzionalità di intelligenza artificiale in dispositivi piccoli e a basso consumo energetico, espandendo la portata di ML a nuovi domini.

La progressione da Cloud a Edge a TinyML riflette una tendenza più ampia nell'informatica verso un'elaborazione più distribuita e localizzata. Questa evoluzione è guidata dalla necessità di tempi di risposta più rapidi, migliore privacy, ridotto utilizzo della larghezza di banda e capacità di operare in ambienti con connettività limitata o assente.
Figura 2.2 illustra le principali differenze tra Cloud ML, Edge ML e TinyML in termini di hardware, latenza, connettività, requisiti di alimentazione e complessità del modello. Passando da Cloud a Edge a TinyML, assistiamo a una drastica riduzione delle risorse disponibili, che presenta sfide significative per l'implementazione di modelli di apprendimento automatico sofisticati. Questa disparità di risorse diventa particolarmente evidente quando si tenta di implementare modelli di deep learning su microcontrollori, la piattaforma hardware principale per TinyML. Questi piccoli dispositivi hanno capacità di memoria e archiviazione fortemente limitate, che sono spesso insufficienti per i modelli di deep learning convenzionali. Impareremo a mettere queste cose in prospettiva in questo capitolo.

	Cloud AI (NVIDIA V100)	➡	Mobile AI (iPhone 11)	➡	Tiny AI (STM32F746)		ResNet-50	MobileNetV2	MobileNetV2 (int8)
Memory	16 GB	4× →	4 GB	3100× →	**320 kB**	← gap →	7.2 MB	6.8 MB	1.7 MB
Storage	TB~PB	1000× →	>64 GB	64000× →	**1 MB**	← gap →	102MB	13.6 MB	3.4 MB

Figura 2.2: Dalle GPU cloud ai microcontrollori: Navigazione nel panorama della memoria e dell'archiviazione tra dispositivi di elaborazione. Fonte: (J. Lin et al. 2023)

2.2 Cloud ML

Cloud ML sfrutta potenti server nel cloud per il training e l'esecuzione di modelli ML complessi e di grandi dimensioni e si basa sulla connettività Internet. Figura 2.3 fornisce una panoramica delle capacità di Cloud ML che discuteremo più in dettaglio in questa sezione.

Figura 2.3: Panoramica della sezione per Cloud ML.

2.2.1 Caratteristiche

Definizione di Cloud ML

Il Cloud Machine Learning (Cloud ML) è un sottocampo del machine learning che sfrutta la potenza e la scalabilità dell'infrastruttura di cloud computing per sviluppare, addestrare e distribuire modelli di machine learning. Utilizzando le vaste risorse computazionali disponibili nel cloud, Cloud ML consente la gestione efficiente di set di dati su larga scala e algoritmi di machine learning complessi.

Infrastruttura Centralizzata

Una delle caratteristiche principali di Cloud ML è la sua infrastruttura centralizzata. Figura 2.4 illustra questo concetto con un esempio dal data center Cloud TPU di Google. I provider di servizi cloud offrono una piattaforma virtuale composta da server ad alta capacità, soluzioni di storage espansive e architetture di rete robuste, tutte ospitate in data center distribuiti in tutto il mondo. Come mostrato nella figura, queste strutture centralizzate possono essere di grandi dimensioni, ospitando file e file di hardware specializzato. Questa configurazione centralizzata consente la messa in comune e la gestione efficiente delle risorse computazionali, semplificando la scalabilità dei progetti di machine learning in base alle esigenze.

Figura 2.4: Data center Cloud TPU presso Google. Fonte: Google.

Elaborazione Dati Scalabile e Addestramento dei Modelli
Il Cloud ML eccelle nella sua capacità di elaborare e analizzare enormi volumi di dati. L'infrastruttura centralizzata è progettata per gestire calcoli complessi e attività di model training che richiedono una notevole potenza di calcolo. Sfruttando la scalabilità del cloud, i modelli di apprendimento automatico possono essere addestrati su grandi quantità di dati, con conseguente miglioramento delle capacità di apprendimento e delle prestazioni predittive.

Deployment Flessibile e Accessibilità
Un altro vantaggio di Cloud ML è la flessibilità che offre in termini di deployment distribuzione e accessibilità. Una volta che un modello di machine learning è stato addestrato e convalidato, può essere facilmente distribuito e reso accessibile agli utenti tramite servizi basati su cloud. Ciò consente un'integrazione perfetta delle funzionalità di apprendimento automatico in varie applicazioni e servizi, indipendentemente dalla posizione o dal dispositivo dell'utente.

Collaborazione e Condivisione delle Risorse
Il Cloud ML promuove la collaborazione e la condivisione delle risorse tra team e organizzazioni. La natura centralizzata dell'infrastruttura cloud consente a più utenti di accedere e lavorare contemporaneamente sugli stessi progetti di apprendimento automatico. Questo approccio collaborativo facilita la condivisione delle conoscenze, accelera il processo di sviluppo e ottimizza l'utilizzo delle risorse.

Efficacia dei Costi e Scalabilità
Sfruttando il modello di prezzo "pay-as-you-go" offerto dai provider di servizi cloud, Cloud ML consente alle organizzazioni di evitare i costi iniziali associati alla creazione e alla manutenzione della propria infrastruttura di machine learning. La capacità di aumentare o diminuire le risorse in base alla domanda garantisce economicità e flessibilità nella gestione dei progetti di apprendimento automatico.
Il Cloud ML ha rivoluzionato il modo in cui ci si approccia all'apprendimento automatico, rendendolo più accessibile, scalabile ed efficiente. Ha aperto nuove possibilità per le organizzazioni di sfruttare la potenza dell'apprendimento automatico senza la necessità di investimenti significativi in hardware e infrastruttura.

2.2.2 Vantaggi

Il Cloud ML offre diversi vantaggi significativi che lo rendono una scelta potente per i progetti di apprendimento automatico:

Immensa Potenza di Calcolo
Uno dei principali vantaggi del Cloud ML è la sua capacità di fornire vaste risorse di calcolo. L'infrastruttura cloud è progettata per gestire algoritmi complessi ed elaborare grandi set di dati in modo efficiente. Ciò è particolarmente vantaggioso per i modelli di apprendimento automatico che richiedono una notevole potenza di calcolo, come reti di deep learning o modelli addestrati su enormi set di dati. Sfruttando le capacità di calcolo del cloud, le organizzazioni possono superare i limiti delle configurazioni hardware locali e ridimensionare i loro progetti di apprendimento automatico per soddisfare requisiti esigenti.

Scalabilità Dinamica
Il Cloud ML offre scalabilità dinamica, consentendo alle organizzazioni di adattarsi facilmente alle mutevoli esigenze di calcolo. Man mano che il volume dei dati aumenta o la complessità dei modelli di apprendimento automatico aumenta, l'infrastruttura cloud può essere ridimensionata senza problemi verso l'alto o verso il basso per adattarsi a questi cambiamenti. Questa flessibilità garantisce prestazioni costanti e consente alle organizzazioni di gestire carichi di lavoro variabili senza la necessità di ingenti investimenti hardware. Col Cloud ML, le risorse possono essere allocate su richiesta, fornendo una soluzione conveniente ed efficiente per la gestione di progetti di machine learning.

Accesso a Strumenti e Algoritmi Avanzati
Le piattaforme Cloud ML forniscono accesso a un'ampia gamma di strumenti e algoritmi avanzati specificamente progettati per l'apprendimento automatico. Questi strumenti spesso includono librerie, framework e API predefiniti che semplificano lo sviluppo e l'implementazione di modelli di apprendimento automatico. Gli sviluppatori possono sfruttare queste risorse per accelerare la creazione, il training e l'ottimizzazione di modelli sofisticati. Utilizzando gli ultimi progressi negli algoritmi e nelle tecniche di apprendimento automatico, le organizzazioni possono rimanere all'avanguardia dell'innovazione e ottenere risultati migliori nei loro progetti di apprendimento automatico.

Ambiente Collaborativo
Il Cloud ML promuove un ambiente collaborativo che consente ai team di lavorare insieme senza problemi. La natura centralizzata dell'infrastruttura cloud consente a più utenti di accedere e contribuire agli stessi progetti di apprendimento automatico contemporaneamente. Questo approccio collaborativo facilita la condivisione delle conoscenze, promuove la collaborazione interfunzionale e accelera lo sviluppo e l'iterazione dei modelli di apprendimento automatico. I team possono condividere facilmente codice, set di dati e risultati, consentendo una collaborazione efficiente e guidando l'innovazione in tutta l'organizzazione.

Efficacia in Termini di Costi
L'adozione del Cloud ML può essere una soluzione conveniente per le organizzazioni, soprattutto rispetto alla creazione e alla manutenzione di un'infrastruttura di apprendimento automatico in sede. I provider di servizi cloud offrono modelli di prezzo flessibili, come piani pay-as-you-go o basati su abbonamento, consentendo alle organizzazioni di pagare solo per le risorse che consumano. Ciò elimina la necessità di investimenti di capitale iniziali in hardware e infrastruttura, riducendo il costo complessivo dell'implementazione di progetti di apprendimento automatico. Inoltre, la scalabilità di Cloud ML garantisce che le

organizzazioni possano ottimizzare l'utilizzo delle risorse ed evitare l'eccesso di provisioning [fornitura], migliorando ulteriormente l'efficienza in termini di costi.
I vantaggi di Cloud ML, tra cui l'immensa potenza di calcolo, la scalabilità dinamica, l'accesso a strumenti e algoritmi avanzati, l'ambiente collaborativo e la convenienza, lo rendono una scelta interessante per le organizzazioni che desiderano sfruttare il potenziale del machine learning. Sfruttando le capacità del cloud, le organizzazioni possono accelerare le proprie iniziative di machine learning, guidare l'innovazione e ottenere un vantaggio competitivo nell'attuale panorama basato sui dati.

2.2.3 Sfide

Sebbene il Cloud ML offra numerosi vantaggi, presenta anche alcune sfide che le organizzazioni devono considerare:

Problemi di Latenza
Una delle principali sfide del Cloud ML è il potenziale di problemi della latenza, in particolare nelle applicazioni che richiedono risposte in tempo reale. Poiché i dati devono essere inviati dall'origine dei dati ai server cloud centralizzati per l'elaborazione e quindi di nuovo all'applicazione, potrebbero verificarsi ritardi dovuti alla trasmissione in rete. Questa latenza può rappresentare un notevole svantaggio in scenari sensibili al fattore tempo, come veicoli autonomi, rilevamento delle frodi in tempo reale o sistemi di controllo industriale, in cui è fondamentale prendere decisioni immediate. Gli sviluppatori devono progettare attentamente i propri sistemi per ridurre al minimo la latenza e garantire tempi di risposta accettabili.

Problemi di Sicurezza e Privacy dei Dati
La centralizzazione dell'elaborazione e dell'archiviazione dei dati nel cloud può sollevare preoccupazioni sulla privacy e sulla sicurezza dei dati. Quando i dati sensibili vengono trasmessi e archiviati in data center remoti, diventano vulnerabili a potenziali attacchi informatici e accessi non autorizzati. I data center cloud possono diventare obiettivi interessanti per gli hacker che cercano di sfruttare le vulnerabilità e ottenere l'accesso a informazioni preziose. Le organizzazioni devono investire in misure di sicurezza robuste, come crittografia, controlli di accesso e monitoraggio continuo, per proteggere i propri dati nel cloud. Anche la conformità alle normative sulla privacy dei dati, come GDPR o HIPAA, diventa una considerazione critica quando si gestiscono dati sensibili nel cloud.

Considerazioni sui Costi
Con l'aumento delle esigenze di elaborazione dei dati, i costi associati all'utilizzo dei servizi cloud possono aumentare. Mentre il Cloud ML offre scalabilità e flessibilità, le organizzazioni che gestiscono grandi volumi di dati potrebbero dover affrontare costi crescenti man mano che consumano più risorse cloud. Il modello di prezzo pay-as-you-go dei servizi cloud implica che i costi possono aumentare rapidamente, soprattutto per attività ad alta intensità di elaborazione come l'addestramento e l'inferenza dei modelli. Le organizzazioni devono monitorare e ottimizzare attentamente l'utilizzo del cloud per garantirne la convenienza. Potrebbero dover prendere in considerazione strategie come la compressione dei dati, la progettazione efficiente degli algoritmi e l'ottimizzazione dell'allocazione delle risorse per ridurre al minimo i costi pur ottenendo le prestazioni desiderate.

Dipendenza dalla Connettività Internet
Il Cloud ML si basa su una connettività Internet stabile e affidabile per funzionare in modo efficace. Poiché i dati devono essere trasmessi da e verso il cloud, eventuali interruzioni o limitazioni nella connettività di rete possono influire sulle prestazioni e sulla disponibilità del sistema di apprendimento automatico. Questa dipendenza dalla connettività Internet può

rappresentare un problema in scenari in cui l'accesso alla rete è limitato, inaffidabile o costoso. Le organizzazioni devono garantire un'infrastruttura di rete solida e considerare meccanismi di "failover" o capacità offline per mitigare l'impatto dei problemi di connettività.

Vendor Lock-In
Quando si adotta il Cloud ML, le organizzazioni spesso diventano dipendenti dagli strumenti, dalle API e dai servizi specifici forniti dal fornitore cloud prescelto. Questo vendor lock-in [blocco da fornitore] può rendere difficile cambiare fornitore o migrare verso piattaforme diverse in futuro. Le organizzazioni possono affrontare sfide in termini di portabilità, interoperabilità e costi quando prendono in considerazione un cambiamento nel loro fornitore di Cloud ML. È importante valutare attentamente le offerte del fornitore, considerare obiettivi strategici a lungo termine e pianificare potenziali scenari di migrazione per ridurre al minimo i rischi associati al vendor lock-in.
Affrontare queste sfide richiede un'attenta pianificazione, progettazione architettonica e strategie di mitigazione del rischio. Le organizzazioni devono soppesare i vantaggi del Cloud ML rispetto ai potenziali problemi e prendere decisioni informate in base ai loro requisiti specifici, alla sensibilità dei dati e agli obiettivi aziendali. Affrontando proattivamente queste sfide, le organizzazioni possono sfruttare efficacemente la potenza del Cloud ML garantendo al contempo la privacy dei dati, la sicurezza, l'economicità e l'affidabilità complessiva del sistema.

2.2.4 Casi d'Uso di Esempio

Il Cloud ML ha trovato ampia adozione in vari domini, rivoluzionando il modo in cui le aziende operano e gli utenti interagiscono con la tecnologia. Esploriamo alcuni esempi notevoli del Cloud ML in azione:

Assistenti Virtuali
Il Cloud ML svolge un ruolo cruciale nel potenziamento di assistenti virtuali come Siri e Alexa. Questi sistemi sfruttano le immense capacità computazionali del cloud per elaborare e analizzare gli input vocali in tempo reale. Sfruttando la potenza dell'elaborazione del linguaggio naturale e degli algoritmi di apprendimento automatico, gli assistenti virtuali possono comprendere le domande degli utenti, estrarre informazioni rilevanti e generare risposte intelligenti e personalizzate. La scalabilità e la potenza di elaborazione del cloud consentono a questi assistenti di gestire un vasto numero di interazioni utente contemporaneamente, offrendo un'esperienza utente fluida e reattiva.

Sistemi di Raccomandazione Commerciali
Il Cloud ML costituisce la spina dorsale dei sistemi di raccomandazione avanzati utilizzati da piattaforme come Netflix e Amazon. Questi sistemi sfruttano la capacità del cloud di elaborare e analizzare enormi set di dati per scoprire pattern, preferenze e comportamenti degli utenti. Sfruttando il filtraggio collaborativo e altre tecniche di apprendimento automatico, i sistemi di raccomandazione possono offrire contenuti personalizzati o suggerimenti di prodotti su misura per gli interessi di ciascun utente. La scalabilità del cloud consente a questi sistemi di aggiornare e perfezionare continuamente le proprie raccomandazioni in base alla quantità sempre crescente di dati utente, migliorandone il coinvolgimento e la soddisfazione.

Rilevamento delle Frodi
Nel settore finanziario, il Cloud ML ha rivoluzionato i sistemi di rilevamento delle frodi. Sfruttando la potenza di calcolo del cloud, questi sistemi possono analizzare grandi quantità di dati transazionali in tempo reale per identificare potenziali attività fraudolente. Gli algoritmi di apprendimento automatico addestrati su pattern di frode storici possono rilevare anomalie

e comportamenti sospetti, consentendo agli istituti finanziari di adottare misure proattive per prevenire le frodi e ridurre al minimo le perdite finanziarie. La capacità del cloud di elaborare e archiviare grandi volumi di dati lo rende una piattaforma ideale per implementare sistemi di rilevamento delle frodi robusti e scalabili.

Esperienze Utente Personalizzate
Il Cloud ML è profondamente integrato nelle nostre esperienze online, plasmando il modo in cui interagiamo con le piattaforme digitali. Dagli annunci personalizzati sui feed dei social media alle funzionalità di testo predittivo nei servizi di posta elettronica, il Cloud ML alimenta algoritmi intelligenti che migliorano il coinvolgimento e la praticità dell'utente. Consente ai siti di e-commerce di consigliare prodotti in base alla cronologia di navigazione e acquisto di un utente, ottimizza i motori di ricerca per fornire risultati accurati e pertinenti e automatizza il tagging e la categorizzazione delle foto su piattaforme come Facebook. Sfruttando le risorse di calcolo del cloud, questi sistemi possono apprendere e adattarsi continuamente alle preferenze dell'utente, offrendo un'esperienza utente più intuitiva e personalizzata.

Sicurezza e Rilevamento delle Anomalie
Il Cloud ML svolge un ruolo nel rafforzare la sicurezza dell'utente alimentando i sistemi di rilevamento delle anomalie. Questi sistemi monitorano costantemente le attività dell'utente e i log di sistema per identificare pattern insoliti o comportamenti sospetti. Analizzando grandi quantità di dati in tempo reale, gli algoritmi Cloud ML possono rilevare potenziali minacce informatiche, come tentativi di accesso non autorizzati, infezioni da malware o violazioni dei dati. La scalabilità e la potenza di elaborazione del cloud consentono a questi sistemi di gestire la crescente complessità e il volume dei dati di sicurezza, fornendo un approccio proattivo alla protezione di utenti e sistemi da potenziali minacce.

2.3 Edge ML

2.3.1 Caratteristiche

Definizione di Edge ML
L'Edge Machine Learning (Edge ML) esegue algoritmi di apprendimento automatico direttamente sui dispositivi endpoint o più vicini al luogo in cui vengono generati i dati anziché affidarsi a server cloud centralizzati. Questo approccio avvicina l'elaborazione alla fonte dei dati, riducendo la necessità di inviarne grandi volumi sulle reti, con conseguente riduzione della latenza e miglioramento della privacy dei dati. Figura 2.5 fornisce una panoramica di questa sezione.

Figura 2.5: Panoramica della sezione per Edge ML.

Elaborazione Dati Decentralizzata

In Edge ML, l'elaborazione dei dati avviene in modo decentralizzato, come illustrato in Figura 2.6. Invece di inviare dati a server remoti, i dati vengono elaborati localmente su dispositivi come smartphone, tablet o dispositivi Internet of Things (IoT). La figura mostra vari esempi di questi dispositivi edge, tra cui dispositivi indossabili, sensori industriali ed elettrodomestici intelligenti. Questa elaborazione locale consente ai dispositivi di prendere decisioni rapide in base ai dati che raccolgono senza affidarsi pesantemente alle risorse di un server centrale.

Figura 2.6: Esempi di Edge ML. Fonte: Edge Impulse.

Archiviazione e Calcolo dei Dati Locali
L'archiviazione e il calcolo dei dati locali sono caratteristiche chiave di Edge ML. Questa configurazione garantisce che i dati possano essere archiviati e analizzati direttamente sui dispositivi, mantenendo così la privacy dei dati e riducendo la necessità di una connettività Internet costante. Inoltre, questo spesso porta a un calcolo più efficiente, poiché i dati non devono percorrere lunghe distanze e i calcoli vengono eseguiti con una comprensione più consapevole del contesto locale, che a volte può portare ad analisi più approfondite.

2.3.2 Vantaggi

Latenza Ridotta
Uno dei principali vantaggi di Edge ML è la significativa riduzione della latenza rispetto al Cloud ML. Questa ridotta latenza può essere un vantaggio fondamentale in situazioni in cui i millisecondi contano, come nei veicoli autonomi, dove un rapido processo decisionale può fare la differenza tra sicurezza e incidente.

Privacy dei Dati Migliorata
Edge ML offre anche una migliore privacy dei dati, poiché i dati vengono principalmente archiviati ed elaborati localmente. Ciò riduce al minimo il rischio di violazioni dei dati, più comuni nelle soluzioni di archiviazione dati centralizzate. Le informazioni sensibili possono essere mantenute più sicure, poiché non vengono inviate su reti che potrebbero essere intercettate.

Minore Utilizzo della Larghezza di Banda
Operare più vicino alla fonte dei dati significa che meno dati devono essere inviati sulle reti, riducendo l'utilizzo della larghezza di banda. Ciò può comportare risparmi sui costi e guadagni di efficienza, soprattutto in ambienti in cui la larghezza di banda è limitata o costosa.

2.3.3 Sfide

Risorse di Calcolo Limitate Rispetto al Cloud ML
Tuttavia, Edge ML presenta le sue sfide. Una delle principali preoccupazioni sono le risorse di calcolo limitate rispetto alle soluzioni basate su cloud. I dispositivi endpoint possono avere una potenza di elaborazione o una capacità di archiviazione diverse rispetto ai server cloud, limitando la complessità dei modelli di apprendimento automatico che possono essere distribuiti.

Complessità nella Gestione dei Nodi Edge
La gestione di una rete di nodi Edge può introdurre complessità, soprattutto per quanto riguarda coordinamento, aggiornamenti e manutenzione. Garantire che tutti i nodi funzionino senza problemi e siano aggiornati con gli algoritmi e i protocolli di sicurezza più recenti può essere una sfida logistica.

Problemi di Sicurezza nei Nodi Edge
Sebbene Edge ML offra una maggiore privacy dei dati, i nodi Edge possono talvolta essere più vulnerabili ad attacchi fisici e informatici. Sviluppare protocolli di sicurezza affidabili che proteggano i dati su ogni nodo senza compromettere l'efficienza del sistema, resta una sfida significativa nell'implementazione di soluzioni Edge ML.

2.3.4 Casi d'Uso di Esempio

Edge ML ha molte applicazioni, dai veicoli autonomi e dalle case intelligenti all'IoT industriale. Questi esempi sono stati scelti per evidenziare scenari in cui l'elaborazione dei dati in tempo reale, la latenza ridotta e la privacy migliorata non sono solo vantaggiose, ma spesso fondamentali per il funzionamento e il successo di queste tecnologie. Dimostrano il

ruolo che Edge ML può svolgere nel guidare i progressi in vari settori, promuovendo l'innovazione e aprendo la strada a sistemi più intelligenti, reattivi e adattabili.

Veicoli Autonomi
I veicoli autonomi sono un esempio lampante del potenziale di Edge ML. Questi veicoli si affidano in larga misura all'elaborazione dei dati in tempo reale per navigare e prendere decisioni. I modelli di apprendimento automatico localizzati aiutano ad analizzare rapidamente i dati da vari sensori per prendere decisioni di guida immediate, garantendo sicurezza e funzionamento regolare.

Case ed Edifici Intelligenti
Edge ML svolge un ruolo cruciale nella gestione efficiente di vari sistemi in case ed edifici intelligenti, dall'illuminazione e dal riscaldamento alla sicurezza. Elaborando i dati localmente, questi sistemi possono funzionare in modo più reattivo e armonioso con le abitudini e le preferenze degli occupanti, creando un ambiente di vita più confortevole.

IoT industriale
L'IoT industriale sfrutta Edge ML per monitorare e controllare processi industriali complessi. Qui, i modelli di apprendimento automatico possono analizzare i dati da numerosi sensori in tempo reale, consentendo la manutenzione predittiva, ottimizzando le operazioni e migliorando le misure di sicurezza. Questa rivoluzione nell'automazione e nell'efficienza industriale sta trasformando la manifattura e la produzione in vari settori.
L'applicabilità di Edge ML è vasta e non si limita a questi esempi. Vari altri settori, tra cui sanità, agricoltura e pianificazione urbana, stanno esplorando e integrando l'Edge ML per sviluppare soluzioni innovative che rispondono alle esigenze e alle sfide del mondo reale, annunciando una nuova era di sistemi intelligenti e interconnessi.

2.4 Tiny ML

2.4.1 Caratteristiche

Definizione di TinyML
TinyML si colloca all'incrocio tra sistemi embedded e apprendimento automatico, rappresentando un campo in rapida crescita che porta algoritmi intelligenti direttamente a microcontrollori e sensori minuscoli. Questi microcontrollori operano con gravi limitazioni di risorse, in particolare per quanto riguarda memoria, archiviazione e potenza di calcolo. Figura 2.7 incapsula gli aspetti chiave di TinyML discussi in questa sezione.

Figura 2.7: Panoramica della sezione per Tiny ML.

Machine Learning On-Device

In TinyML, l'attenzione è rivolta all'apprendimento automatico sul dispositivo. Ciò significa che i modelli di apprendimento automatico vengono distribuiti e addestrati sul dispositivo, eliminando la necessità di server esterni o infrastrutture cloud. Ciò consente a TinyML di abilitare un processo decisionale intelligente proprio dove vengono generati i dati, rendendo possibili approfondimenti e azioni in tempo reale, anche in contesti in cui la connettività è limitata o non disponibile.

Ambienti a Basso Consumo Energetico e con Risorse Limitate

TinyML eccelle in contesti a basso consumo energetico e con risorse limitate. Questi ambienti richiedono soluzioni altamente ottimizzate che funzionino entro le risorse disponibili. Figura 2.8 mostra un kit di dispositivi TinyML di esempio, che illustra la natura compatta di questi sistemi. Questi dispositivi possono solitamente stare nel palmo di una mano o, in alcuni casi, sono persino piccoli come un'unghia. TinyML soddisfa l'esigenza di efficienza tramite algoritmi e modelli specializzati progettati per offrire prestazioni decenti consumando al contempo un'energia minima, garantendo così periodi operativi prolungati, anche in dispositivi alimentati a batteria come quelli mostrati.

Figura 2.8: Esempi di kit di dispositivi TinyML. Fonte: Widening Access to Applied Machine Learning with TinyML.

Esercizio 2.1: TinyML con Arduino

Prepararsi a portare l'apprendimento automatico sui dispositivi più piccoli! Nel mondo dell'apprendimento automatico embedded, TinyML è il luogo in cui i vincoli di risorse incontrano l'ingegnosità. Questo notebook Colab guiderà nella creazione di un modello di riconoscimento dei gesti progettato su una scheda Arduino. Si imparerà come addestrare una piccola ma efficace rete neurale, ottimizzarla per un utilizzo minimo di memoria e distribuirla al proprio microcontrollore. Se si è entusiasti di rendere più intelligenti gli oggetti di uso quotidiano, è qui che si inizia!

Open in Colab

2.4.2 Vantaggi

Latenza Estremamente Bassa

Uno dei vantaggi più importanti di TinyML è la sua capacità di offrire una latenza estremamente bassa. Poiché il calcolo avviene direttamente sul dispositivo, il tempo necessario per inviare dati a server esterni e ricevere una risposta viene eliminato. Ciò è fondamentale nelle applicazioni che richiedono un processo decisionale immediato, consentendo risposte rapide a condizioni mutevoli.

Elevata Sicurezza dei Dati

TinyML migliora intrinsecamente la sicurezza dei dati. Poiché l'elaborazione e l'analisi dei dati avvengono sul dispositivo, il rischio di intercettazione dei dati durante la trasmissione viene praticamente eliminato. Questo approccio localizzato alla gestione dei dati garantisce che le informazioni sensibili rimangano sul dispositivo, rafforzando la sicurezza dei dati dell'utente.

Efficienza Energetica

TinyML opera all'interno di un framework efficiente dal punto di vista energetico, una necessità dati i suoi ambienti con risorse limitate. Utilizzando algoritmi snelli e metodi di calcolo ottimizzati, TinyML garantisce che i dispositivi possano eseguire attività complesse senza esaurire rapidamente la durata della batteria, il che lo rende un'opzione sostenibile per le distribuzioni a lungo termine.

2.4.3 Sfide

Capacità di Calcolo Limitate

Tuttavia, il passaggio a TinyML comporta una serie di ostacoli. La limitazione principale sono le capacità di calcolo limitate dei dispositivi. La necessità di operare entro tali limiti implica che i modelli distribuiti debbano essere semplificati, il che potrebbe influire sull'accuratezza e la complessità delle soluzioni.

Ciclo di Sviluppo Complesso

TinyML introduce anche un ciclo di sviluppo complicato. La creazione di modelli leggeri ed efficaci richiede una profonda comprensione dei principi di apprendimento automatico e competenza nei sistemi embedded. Questa complessità richiede un approccio di sviluppo collaborativo, in cui la competenza multi-dominio è essenziale per il successo.

Ottimizzazione e Compressione del Modello

Una sfida centrale in TinyML è l'ottimizzazione e la compressione del modello. La creazione di modelli di machine learning in grado di operare efficacemente all'interno della memoria limitata e della potenza di calcolo dei microcontrollori richiede approcci innovativi alla

progettazione del modello. Gli sviluppatori si trovano spesso ad affrontare la sfida di trovare un delicato equilibrio e ottimizzare i modelli per mantenere l'efficacia, pur rispettando rigidi vincoli di risorse.

2.4.4 Casi d'Uso di Esempio

Dispositivi Indossabili

Nei dispositivi indossabili, TinyML apre le porte a gadget più intelligenti e reattivi. Dai fitness tracker che offrono feedback in tempo reale sugli allenamenti agli occhiali intelligenti che elaborano dati visivi al volo, TinyML trasforma il modo in cui interagiamo con la tecnologia indossabile, offrendo esperienze personalizzate direttamente dal dispositivo.

Manutenzione Predittiva

Negli ambienti industriali, TinyML svolge un ruolo significativo nella manutenzione predittiva. Implementando algoritmi TinyML su sensori che monitorano lo stato di salute delle apparecchiature, le aziende possono identificare preventivamente potenziali problemi, riducendo i tempi di inattività e prevenendo costosi guasti. L'analisi dei dati in loco garantisce risposte rapide, impedendo potenzialmente a piccoli problemi di diventare problemi gravi.

Rilevamento delle Anomalie

TinyML può essere impiegato per creare modelli di rilevamento delle anomalie che identificano pattern di dati insoliti. Ad esempio, una fabbrica intelligente potrebbe usare TinyML per monitorare i processi industriali e individuare anomalie, aiutando a prevenire incidenti e migliorare la qualità del prodotto. Allo stesso modo, un'azienda di sicurezza potrebbe usare TinyML per monitorare il traffico di rete per pattern insoliti, aiutando a rilevare e prevenire attacchi informatici. TinyML potrebbe monitorare i dati dei pazienti per anomalie nell'assistenza sanitaria, aiutando a rilevare precocemente le malattie e a migliorare il trattamento dei pazienti.

Monitoraggio Ambientale

Nel monitoraggio ambientale, TinyML consente l'analisi dei dati in tempo reale da vari sensori distribuiti sul campo. Questi potrebbero spaziare dal monitoraggio della qualità dell'aria in città al tracciamento della fauna selvatica nelle aree protette. Tramite TinyML, i dati possono essere elaborati localmente, consentendo risposte rapide alle mutevoli condizioni e fornendo una comprensione adeguata dei modelli pattern, cruciale per un processo decisionale informato.

In sintesi, TinyML funge da pioniere nell'evoluzione dell'apprendimento automatico, promuovendo l'innovazione in vari campi portando l'intelligenza direttamente nell'Edge. Il suo potenziale di trasformare la nostra interazione con la tecnologia e il mondo è immenso, promettendo un futuro in cui i dispositivi sono connessi, intelligenti e capaci di prendere decisioni e rispondere in tempo reale.

2.5 Confronto

Mettiamo insieme le diverse varianti di ML che abbiamo esplorato individualmente per una visione completa. Figura 2.9 illustra le relazioni e le sovrapposizioni tra Cloud ML, Edge ML e TinyML utilizzando un diagramma di Venn. Questa rappresentazione visiva evidenzia efficacemente le caratteristiche uniche di ciascun approccio, mostrando anche le aree in comune. Ogni paradigma di ML ha le sue caratteristiche distinte, ma ci sono anche intersezioni in cui questi approcci condividono determinati attributi o capacità. Questo diagramma ci aiuta a capire come queste varianti si relazionano tra loro nel più ampio panorama delle implementazioni di apprendimento automatico.

Figura 2.9: Diagramma di Venn ML. Fonte: arXiv

Per un confronto più dettagliato di queste varianti di ML, possiamo fare riferimento a Tabella 2.1. Questa tabella offre un'analisi completa di Cloud ML, Edge ML e TinyML in base a varie caratteristiche e aspetti. Esaminando queste diverse caratteristiche una accanto all'altra, otteniamo una prospettiva più chiara sui vantaggi unici e sui fattori distintivi di ogni approccio. Questo confronto dettagliato, combinato con la panoramica visiva fornita dal diagramma di Venn, aiuta a prendere decisioni informate in base alle esigenze e ai vincoli specifici di una determinata applicazione o progetto.

Tabella 2.1: Confronto degli aspetti delle funzionalità tra Cloud ML, Edge ML e TinyML.

Aspetto	Cloud ML	Edge ML	TinyML
Ubicazione della elaborazione	Server centralizzati (Data Center)	Dispositivi locali (più vicini alle fonti di dati)	Sul dispositivo (microcontrollori, sistemi embedded)
Latenza	Alta (dipende dalla connettività Internet)	Moderata (latenza ridotta rispetto a Cloud ML)	Bassa (elaborazione immediata senza ritardo di rete)
Privacy dei dati	Moderata (dati trasmessi tramite reti)	Alta (i dati rimangono sulle reti locali)	Molto alta (dati elaborati sul dispositivo, non trasmessi)
Potenza di calcolo	Alta (usa una potente infrastruttura del data center)	Moderata (utilizza le capacità del dispositivo locale)	Bassa (limitata alla potenza del sistema embedded)
Consumo energetico	Alto (i data center consumano molta energia)	Moderato (meno dei data center, più di TinyML)	Basso (alta efficienza energetica, progettato per bassi consumi)
Scalabilità	Alto (facile da scalare con risorse server aggiuntive)	Moderato (dipende dalle capacità del dispositivo locale)	Basso (limitato dalle risorse hardware del dispositivo)
Costo	Alto (costi ricorrenti per l'uso del server, manutenzione)	Variabile (dipende dalla complessità della configurazione locale)	Basso (principalmente costi iniziali per i componenti hardware)
Connettività	Alto (richiede una connettività Internet stabile)	Basso (può funzionare con connettività intermittente)	Molto basso (può funzionare senza alcuna connettività di rete)

Elaborazione in tempo reale	Moderata (può essere influenzata dalla latenza di rete)	Alta (capace di elaborazione in tempo reale localmente)	Molto alta (elaborazione immediata con latenza minima)
Esempi di applicazione	Analisi di Big Data, Assistenti virtuali	Veicoli autonomi, Case intelligenti	Dispositivi indossabili, Reti di sensori
Complessità	Da moderata ad alta (richiede conoscenza del cloud computing)	Moderata (richiede conoscenza della configurazione della rete locale)	Da moderata ad alta (richiede competenza nei sistemi embedded)

2.6 Conclusione

In questo capitolo, abbiamo offerto una panoramica in evoluzione dell'apprendimento automatico, che copre i paradigmi cloud, edge e tiny ML. L'apprendimento automatico basato su cloud sfrutta le immense risorse computazionali delle piattaforme cloud per abilitare modelli potenti e accurati, ma presenta delle limitazioni, tra cui problemi di latenza e privacy. Edge ML mitiga queste limitazioni portando l'inferenza direttamente sui dispositivi edge, offrendo una latenza inferiore e ridotte esigenze di connettività. TinyML va oltre, miniaturizzando i modelli ML per eseguirli direttamente su dispositivi con risorse altamente limitate, aprendo una nuova categoria di applicazioni intelligenti.

Ogni approccio ha i suoi compromessi, tra cui complessità del modello, latenza, privacy e costi dell'hardware. Nel tempo, prevediamo la convergenza di questi approcci ML embedded, col pre-training cloud che facilita implementazioni edge e tiny ML più sofisticate. Progressi come l'apprendimento federato e l'apprendimento "on-device" consentiranno ai dispositivi embedded di perfezionare i propri modelli imparando dai dati del mondo reale.

Il panorama ML embedded si sta evolvendo rapidamente ed è pronto a consentire applicazioni intelligenti su un ampio spettro di dispositivi e casi d'uso. Questo capitolo funge da "istantanea" dello stato attuale del ML embedded. Man mano che algoritmi, hardware e connettività continuano a migliorare, possiamo aspettarci che i dispositivi embedded di tutte le dimensioni diventino sempre più capaci, sbloccando nuove applicazioni trasformative per l'intelligenza artificiale.

2.7 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Stiamo lavorando costantemente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

ⓘ Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Embedded Systems Overview.
- Embedded Computer Hardware.
- Embedded I/O.
- Embedded systems software.
- Embedded ML software.
- Embedded Inference.
- TinyML on Microcontrollers.
- TinyML as a Service (TinyMLaaS):
  - TinyMLaaS: Introduction.
  - TinyMLaaS: Design Overview.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- *Prossimamente*.

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

3. Avvio al Deep Learning

DALL·E 3 Prompt: Foto di un'aula classica con una grande lavagna che domina una parete. I disegni a gesso mostrano una rete neurale profonda dettagliata con diversi livelli nascosti e ogni nodo e connessione è etichettato con precisione con il gesso bianco. Il pavimento in legno rustico e le pareti in mattoni creano un contrasto con i concetti moderni. Intorno alla stanza, poster incorniciati sottolineano i temi del deep learning: reti convoluzionali, trasformatori, neuroni, funzioni di attivazione e altro ancora.

Questa sezione funge da introduzione al deep learning, fornendo ai professionisti dei sistemi il contesto essenziale e le conoscenze fondamentali necessarie per implementare efficacemente soluzioni di deep learning. Invece di addentrarci in profondità teoriche, ci concentriamo su concetti chiave, architetture e considerazioni pratiche rilevanti per l'implementazione dei sistemi. Iniziamo con una panoramica dell'evoluzione del deep learning e del suo particolare significato nei sistemi di intelligenza artificiale embedded. Concetti fondamentali come le reti neurali vengono introdotti con un'enfasi sulle considerazioni di implementazione piuttosto che sulle basi matematiche.
L'introduzione esplora le principali architetture di deep learning da una prospettiva di sistema, esaminandone le implicazioni pratiche e i requisiti di risorse. Confrontiamo inoltre il deep learning con gli approcci tradizionali di machine learning, aiutando i lettori a fare scelte architettoniche informate basate su vincoli di sistema del mondo reale. Questa panoramica di alto livello definisce il contesto per le tecniche e le ottimizzazioni più dettagliate incentrate sui sistemi trattate nei capitoli successivi.

Obiettivi dell'Apprendimento

- Comprendere i concetti di base e le definizioni delle reti neurali profonde.
- Riconoscere che esistono diverse architetture di modelli di deep learning.
- Confronto tra deep learning e approcci di machine learning [apprendimento automatico] tradizionali in varie dimensioni.
- Acquisire gli elementi concettuali di base per approfondire le tecniche e le applicazioni avanzate del deep learning.

3.1 Introduzione

3.1.1 Definizione e Importanza

Il deep learning, un'area specializzata nell'apprendimento automatico e nell'intelligenza artificiale (IA), utilizza algoritmi modellati sulla struttura e la funzione del cervello umano, noti come reti neurali artificiali. Questo campo è un elemento fondamentale nell'IA, che guida il progresso in diversi settori come la visione artificiale, l'elaborazione del linguaggio naturale e i veicoli a guida autonoma. La sua importanza nei sistemi di IA embedded è evidenziata dalla sua capacità di gestire calcoli e previsioni intricati, ottimizzando le risorse limitate nelle impostazioni embedded.

La Figura 3.1 fornisce una rappresentazione visiva di come il deep learning si inserisce nel contesto più ampio dell'IA e del "machine learning" [apprendimento automatico]. Il diagramma illustra lo sviluppo cronologico e la relativa segmentazione di questi tre campi interconnessi, mostrando il deep learning come un sottoinsieme specializzato dell'apprendimento automatico, che a sua volta è un sottoinsieme dell'IA.

Figura 3.1: Il diagramma illustra l'intelligenza artificiale come campo onnicomprensivo che comprende tutti i metodi computazionali che imitano le funzioni cognitive umane. Il Machine learning [apprendimento automatico] è un sottoinsieme dell'IA che include algoritmi in grado di apprendere dai dati. Il deep learning, un ulteriore sottoinsieme del ML, coinvolge specificamente reti neurali in grado di apprendere pattern [schemi] più complessi in grandi volumi di dati. Fonte: NVIDIA.

Come mostrato nella figura, l'IA rappresenta il campo sovraordinato, che comprende tutti i metodi computazionali che imitano le funzioni cognitive umane. Il machine learning è mostrato come un sottoinsieme dell'IA che include algoritmi in grado di apprendere dai dati. Il deep learning, il sottoinsieme più piccolo nel diagramma, coinvolge specificamente reti neurali in grado di apprendere modelli più complessi da grandi volumi di dati.

3.1.2 Breve Storia del Deep Learning

L'idea del deep learning ha origine nelle prime reti neurali artificiali. Ha vissuto diversi cicli di interesse, a partire dall'introduzione del Perceptron negli anni '50 (Rosenblatt 1957), seguita dall'invenzione degli algoritmi di backpropagation negli anni '80 (Rumelhart, Hinton, e Williams 1986).

Il termine “deep learning” è diventato importante negli anni 2000, caratterizzato da progressi nella potenza di calcolo e nell’accessibilità dei dati. Traguardi importanti includono l’addestramento di successo di reti profonde come AlexNet (Krizhevsky, Sutskever, e Hinton 2012) da parte di Geoffrey Hinton, una figura di spicco nell’intelligenza artificiale, e il rinnovato focus sulle reti neurali come strumenti efficaci per l’analisi e la modellazione dei dati.

Il deep learning ha recentemente registrato una crescita esponenziale, trasformando vari settori. La Figura 3.2 illustra questa notevole progressione, evidenziando due tendenze chiave nel settore. Innanzitutto, il grafico mostra che la crescita computazionale ha seguito un modello di raddoppio di 18 mesi dal 1952 al 2010. Questa tendenza ha poi accelerato drasticamente fino a un ciclo di raddoppio di 6 mesi dal 2010 al 2022, indicando un balzo significativo nelle capacità computazionali.

In secondo luogo, la figura raffigura l’emergere di modelli su larga scala tra il 2015 e il 2022. Questi modelli sono apparsi da 2 a 3 ordini di grandezza più veloci rispetto alla tendenza generale, seguendo un ciclo di raddoppio di 10 mesi ancora più aggressivo. Questo rapido ridimensionamento delle dimensioni del modello rappresenta un cambiamento di paradigma nelle capacità di deep learning.

Figura 3.2: Crescita dei modelli di deep learning.

Molteplici fattori hanno contribuito a questa impennata, tra cui i progressi nella potenza computazionale, l’abbondanza di big data e i miglioramenti nei progetti algoritmici. In primo luogo, la crescita delle capacità computazionali, in particolare l’arrivo delle Graphics Processing Units (GPU) [unità di elaborazione grafica] e delle Tensor Processing Units (TPU) [unità di elaborazione tensoriale] (N. P. Jouppi et al. 2017a), ha accelerato notevolmente i tempi di training e inferenza dei modelli di apprendimento profondo. Questi miglioramenti hardware hanno consentito la costruzione e il training di reti più complesse e profonde di quanto fosse possibile negli anni precedenti.

In secondo luogo, la rivoluzione digitale ha prodotto una grande quantità di big data, offrendo materiale ricco da cui i modelli di deep learning possono imparare e distinguersi in attività quali il riconoscimento di immagini e parlato, la traduzione linguistica e il gioco. I grandi set di dati etichettati sono stati fondamentali per perfezionare e distribuire con successo applicazioni di deep learning in contesti reali.

Inoltre, le collaborazioni e gli sforzi open source hanno alimentato una comunità dinamica di ricercatori e professionisti, accelerando i progressi nelle tecniche di deep learning. Innovazioni come il “deep reinforcement learning”, il “transfer learning” e l’intelligenza artificiale generativa hanno ampliato la portata di ciò che è realizzabile col deep learning, aprendo nuove possibilità in vari settori, tra cui sanità, finanza, trasporti e intrattenimento. Le organizzazioni di tutto il mondo riconoscono il potenziale trasformativo del deep learning e investono molto in ricerca e sviluppo per sfruttare le sue capacità nel fornire soluzioni innovative, ottimizzare le operazioni e creare nuove opportunità di business. Mentre il deep learning continua la sua traiettoria ascendente, è destinato a ridefinire il modo in cui interagiamo con la tecnologia, migliorando la praticità, la sicurezza e la connettività nelle nostre vite.

3.1.3 Applicazioni del Deep Learning

Il deep learning è ampiamente utilizzato in numerosi settori oggi, con il suo impatto trasformativo evidente in vari settori, come illustrato in Figura 3.3. Nella finanza, alimenta la previsione del mercato azionario, la valutazione del rischio e il rilevamento delle frodi, guidando le strategie di investimento e migliorando le decisioni finanziarie. Il marketing sfrutta il deep learning per la segmentazione e la personalizzazione dei clienti, consentendo pubblicità altamente mirate e l’ottimizzazione dei contenuti in base all’analisi del comportamento dei consumatori. Nella produzione, semplifica i processi e migliora il controllo di qualità, consentendo alle aziende di aumentare la produttività e ridurre al minimo gli sprechi. L’assistenza sanitaria trae vantaggio dal deep learning nella diagnosi, nella pianificazione del trattamento e nel monitoraggio dei pazienti, salvando potenzialmente vite umane attraverso migliori previsioni mediche.

Figura 3.3: Applicazioni, vantaggi e implementazioni del deep learning in vari settori, tra cui finanza, marketing, produzione e assistenza sanitaria. Fonte: Leeway Hertz

Oltre a questi settori principali, il deep learning migliora i prodotti e i servizi di tutti i giorni. Netflix lo utilizza per rafforzare i suoi sistemi di raccomandazione, fornendo agli utenti più raccomandazioni personalizzate. Google ha migliorato notevolmente il suo servizio di traduzione, gestendo ora oltre 100 lingue con maggiore accuratezza, come evidenziato nei suoi recenti progressi. I veicoli autonomi di aziende come Waymo, Cruise e Motional sono diventati realtà grazie al deep learning nel loro perception system. Inoltre, Amazon impiega il

deep learning edge nei dispositivi Alexa per attività come individuazione di parole chiave. Queste applicazioni dimostrano come il machine learning spesso predice ed elabora le informazioni con maggiore accuratezza e velocità rispetto agli esseri umani, rivoluzionando vari aspetti della nostra vita quotidiana.

3.1.4 Rilevanza per l'IA Embedded

L'IA embedded, l'integrazione di algoritmi di intelligenza artificiale direttamente nei dispositivi hardware, trae naturalmente vantaggio dalle capacità del deep learning. La combinazione di algoritmi di deep learning e sistemi embedded ha gettato le basi per dispositivi intelligenti e autonomi in grado di analisi avanzate on-device [sul dispositivo]. Il deep learning aiuta a estrarre pattern e informazioni complesse dai dati di input, il che è essenziale nello sviluppo di sistemi embedded intelligenti, dagli elettrodomestici ai macchinari industriali. Questa collaborazione inaugura una nuova era di dispositivi intelligenti e interconnessi, in grado di apprendere e adattarsi al comportamento dell'utente e alle condizioni ambientali, ottimizzando le prestazioni e offrendo praticità ed efficienza senza precedenti.

3.2 Reti Neurali

Il deep learning trae ispirazione dalle reti neurali del cervello umano per creare pattern decisionali. Questa sezione approfondisce i concetti fondamentali del deep learning, offrendo approfondimenti sugli argomenti più complessi trattati più avanti in questa introduzione.

Le reti neurali fungono da fondamento del deep learning, ispirate alle reti neurali biologiche nel cervello umano per elaborare e analizzare i dati in modo gerarchico. Le reti neurali sono composte da unità di base chiamate perceptron, che sono solitamente organizzate in layer [strati]. Ogni layer è costituito da diversi perceptron e più layer sono impilati per formare l'intera rete. Le connessioni tra questi layer sono definite da insiemi di pesi o parametri che determinano come i dati vengono elaborati mentre fluiscono dall'input all'output della rete. Di seguito, esaminiamo i componenti e le strutture primarie nelle reti neurali.

3.2.1 Perceptron

Il Perceptron è l'unità di base o il nodo che costituisce la base per strutture più complesse. Funziona prendendo più input, ognuno dei quali rappresenta una caratteristica dell'oggetto in analisi, come le caratteristiche di una casa per prevederne il prezzo o gli attributi di una canzone per prevederne la popolarità nei servizi di streaming musicale. Questi input sono indicati come $x_1, x_2, \ldots, x_n$. Un perceptron può essere configurato per eseguire attività di regressione o classificazione. Per la regressione, viene utilizzato l'output numerico effettivo $\hat{y}$. Per la classificazione, l'output dipende dal fatto che $\hat{y}$ superi una determinata soglia. Se $\hat{y}$ supera questa soglia, il perceptron potrebbe restituire una classe (ad esempio, 'yes') e, in caso contrario, un'altra classe (ad esempio, 'no').

Figura 3.4 illustra gli elementi fondamentali di un perceptron, che funge da fondamento per reti neurali più complesse. Un perceptron può essere pensato come un decisore in miniatura, che utilizza i suoi pesi, il sui bias [polarizzazione] e la sua funzione di attivazione per elaborare input e generare output in base ai parametri appresi. Questo concetto costituisce la base per comprendere architetture di reti neurali più complesse, come i perceptron multilayer [multistrato]. In queste strutture avanzate, i layer di perceptron lavorano di concerto, con l'output di ogni layer che funge da input per il layer successivo. Questa disposizione gerarchica crea un modello di deep learning in grado di comprendere e modellare pattern complessi e astratti all'interno dei dati. Impilando queste semplici unità, le reti neurali acquisiscono la capacità di affrontare attività sempre più sofisticate, dal riconoscimento delle immagini all'elaborazione del linguaggio naturale.

Figura 3.4: Perceptron. Concepiti negli anni '50, i perceptron hanno aperto la strada allo sviluppo di reti neurali più complesse e sono stati un elemento fondamentale nel deep learning. Fonte: Wikimedia - Chrislb.

Ciascun input x_i ha un peso corrispondente w_{ij} e il perceptron moltiplica semplicemente ogni input per il suo peso corrispondente. Questa operazione è simile alla regressione lineare, dove l'output intermedio, z, è calcolato come la somma dei prodotti degli input e dei loro pesi:

$$z = \sum(x_i \cdot w_{ij})$$

A questo calcolo intermedio, viene aggiunto un termine di bias b, che consente al modello di adattarsi meglio ai dati spostando la funzione di output lineare verso l'alto o verso il basso. Pertanto, la combinazione lineare intermedia calcolata dal perceptron, incluso il bias, diventa:

$$z = \sum(x_i \cdot w_{ij}) + b$$

Questa forma base di un perceptron può modellare solo relazioni lineari tra input e output. I pattern trovati in natura sono spesso complessi e si estendono oltre le relazioni lineari. Per consentire al perceptron di gestire relazioni non lineari, una funzione di attivazione viene applicata all'output lineare z.

$$\hat{y} = \sigma(z)$$

Figura 3.5 illustra un esempio in cui i dati presentano un pattern non lineare che non potrebbe essere modellato adeguatamente con un approccio lineare. La funzione di attivazione, come la sigmoide, la tanh o la ReLU, trasforma la somma di input lineare in un output non lineare. L'obiettivo principale di questa funzione è introdurre la non linearità nel modello, consentendogli di apprendere ed eseguire attività più sofisticate. Pertanto, l'output finale del perceptron, inclusa la funzione di attivazione, può essere espresso come:

Figura 3.5: Le funzioni di attivazione consentono la modellazione di relazioni non lineari complesse. Fonte: Medium - Sachin Kaushik.

3.2.2 Perceptron Multilayer

I "Multilayer perceptron" (MLP) sono un'evoluzione del modello del perceptron a singolo layer, caratterizzato da più layer di nodi collegati in modo "feedforward". Figura 3.6 fornisce una rappresentazione visiva di questa struttura. Come illustrato nella figura, le informazioni in una rete "feedforward" si muovono in una sola direzione: dal livello di input a sinistra, attraverso i livelli nascosti al centro, fino al livello di output a destra, senza cicli o loop.

Figura 3.6: Perceptron Multilayer. Fonte: Wikimedia - Charlie.

Mentre un singolo perceptron è limitato nella sua capacità di modellare pattern complessi, la vera forza delle reti neurali emerge dall'assemblaggio di più layer. Ciascun layer è costituito da numerosi perceptron che lavorano insieme, consentendo alla rete di catturare relazioni intricate e non lineari all'interno dei dati. Con sufficiente profondità e ampiezza, queste reti

possono approssimare praticamente qualsiasi funzione, indipendentemente da quanto sia complessa.

3.2.3 Processo di Training

Una rete neurale riceve un input, esegue un calcolo e produce una previsione. La previsione è determinata dai calcoli eseguiti all'interno dei set di perceptron trovati tra i layer di input e output. Questi calcoli dipendono principalmente dall'input e dai pesi. Poiché non si ha il controllo sull'input, l'obiettivo durante il training [addestramento] è quello di regolare i pesi in modo tale che l'output della rete fornisca la previsione più accurata.
Il processo di addestramento prevede diversi passaggi chiave, a partire dal passaggio in avanti (forward), in cui i pesi esistenti della rete vengono utilizzati per calcolare l'output per un dato input. Questo output viene poi confrontato con i veri valori target per calcolare un errore, che misura quanto bene la previsione della rete corrisponde al risultato previsto. In seguito, viene eseguito un passaggio all'indietro (backward). Ciò comporta l'utilizzo dell'errore per apportare modifiche ai pesi della rete tramite un processo chiamato "backpropagation". Questa regolazione riduce l'errore nelle previsioni successive. Il ciclo di passaggio forward [in avanti], calcolo dell'errore e passaggio backward [all'indietro] viene ripetuto iterativamente. Questo processo continua finché le previsioni della rete non sono sufficientemente accurate o non viene raggiunto un numero predefinito di iterazioni, riducendo al minimo la "funzione di perdita" utilizzata per misurare l'errore.

3.2.3.1 Forward Pass

Il forward pass è la fase iniziale in cui i dati si spostano attraverso la rete dal livello di input a quello di output, come illustrato in Figura 3.7. All'inizio dell'addestramento, i pesi della rete vengono inizializzati in modo casuale, impostando le condizioni iniziali. Durante il "forward pass", ogni layer esegue calcoli specifici sui dati di input utilizzando questi pesi e il bias, e i risultati vengono poi passati al layer successivo. L'output finale di questa fase è la previsione della rete. Questa "prediction" viene confrontata con i valori target effettivi presenti nel set di dati per calcolare la "loss" [perdita], che può essere considerata come la differenza tra gli output previsti e i valori target. La perdita quantifica le prestazioni della rete in questa fase, fornendo una metrica cruciale per la successiva regolazione dei pesi durante il backward pass.

Figura 3.7: Reti neurali: propagazione forward e backward. Fonte: Linkedin

3.2.3.2 Backward Pass (Backpropagation)

Dopo aver completato il forward pass e calcolato la perdita, che misura quanto le previsioni del modello si discostano dai valori target effettivi, il passo successivo è migliorare le prestazioni del modello regolando i pesi della rete. Poiché non possiamo controllare gli input del modello, la regolazione dei pesi diventa il nostro metodo principale per perfezionare il modello.

Determiniamo come regolare i pesi del nostro modello tramite un algoritmo chiave chiamato "backpropagation". La backpropagation utilizza la perdita calcolata per determinare il gradiente di ciascun peso. Questi gradienti descrivono la direzione e l'entità in cui i pesi devono essere regolati. Regolando i pesi in base a questi gradienti, il modello è meglio posizionato per fare previsioni più vicine ai valori target effettivi nel successivo "forward pass".

Comprendere questi concetti fondamentali apre la strada alla comprensione di architetture e tecniche di deep learning più complesse, favorendo lo sviluppo di applicazioni più sofisticate e produttive, in particolare all'interno di sistemi di intelligenza artificiale embedded.

Video 3.1 and Video 3.2 build upon Video 3.3. Riguardano la "gradient descent" [discesa del gradiente] e la backpropagation nelle reti neurali.

Video 3.1: Gradient descent

https://www.youtube.com/watch?v=IHZwWFHWa-w&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=2

Video 3.2: Backpropagation

https://www.youtube.com/watch?v=Ilg3gGewQ5U&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi&index=3

3.2.4 Architetture dei Modelli

Le architetture di deep learning si riferiscono ai vari approcci strutturati che stabiliscono come i neuroni e i layer sono organizzati e interagiscono nelle reti neurali. Queste architetture si sono evolute per affrontare efficacemente diversi problemi e diversi tipi di dati. Questa sezione fornisce una panoramica di alcune note architetture di deep learning e delle loro caratteristiche.

3.2.4.1 Multilayer Perceptron (MLP)

Gli MLP sono architetture di deep learning di base che comprendono tre layer: uno di input, uno o più layer nascosti e un layer di output. Questi layer sono completamente connessi, il che significa che ogni neurone in uno layer è collegato a ogni neurone nei layer precedenti e successivi. Gli MLP possono modellare funzioni complesse e sono utilizzati in varie attività, come regressione, classificazione e riconoscimento di pattern. La loro capacità di apprendere relazioni non lineari tramite backpropagation li rende uno strumento versatile nel toolkit di deep learning.

Nei sistemi di intelligenza artificiale embedded, gli MLP possono funzionare come modelli compatti per attività più semplici come l'analisi dei dati dei sensori o il riconoscimento di pattern di base, in cui le risorse computazionali sono limitate. La loro capacità di apprendere relazioni non lineari con una complessità relativamente minore li rende una scelta adatta per i sistemi embedded.

Esercizio 3.1: Multilayer Perceptron (MLP)

Abbiamo appena scalfito la superficie delle reti neurali. Ora, proveremo ad applicare questi concetti in esempi pratici. Nei notebook Colab forniti, si esploreranno:

Previsione dei prezzi delle case: Scoprire come le reti neurali possono analizzare i dati sugli alloggi per stimare i valori delle proprietà. Open in Colab

Classificazione delle immagini: Scoprire come creare una rete per comprendere il famoso set di dati di cifre scritte a mano MNIST. Open in Colab

Diagnosi medica nel mondo reale: Usare il deep learning per affrontare l'importante compito della classificazione del cancro al seno. Open in Colab

3.2.4.2 Convolutional Neural Networks (CNNs)

Le CNN [reti neurali convoluzionali] sono utilizzate principalmente in attività di riconoscimento di immagini e video. Questa architettura è composta da due parti principali: la base convoluzionale e i layer completamente connessi. Nella base convoluzionale, i layer convoluzionali filtrano i dati di input per identificare caratteristiche come bordi, angoli e texture [trame]. Dopo ogni layer convoluzionale, è possibile applicare un layer di pooling [raggruppamento] per ridurre le dimensioni spaziali dei dati, diminuendo così il carico computazionale e concentrando le feature estratte. A differenza degli MLP, che trattano le feature di input come entità piatte e indipendenti, le CNN mantengono le relazioni spaziali tra i pixel, rendendole particolarmente efficaci per i dati di immagini e video. Le feature estratte dalla base convoluzionale vengono poi passate ai layer completamente connessi, simili a quelli utilizzati negli MLP, che eseguono la classificazione in base alle feature estratte dai layer di convoluzione. Le CNN si sono dimostrate altamente efficaci nel riconoscimento delle immagini, nel rilevamento di oggetti e in altre applicazioni di visione artificiale.
Video 3.3 spiega come funzionano le reti neurali usando il riconoscimento di cifre scritte a mano come applicazione di esempio. Affronta anche la matematica alla base delle reti neurali.

Video 3.3: Reti MLP & CNN

https://www.youtube.com/embed/aircAruvnKk?si=ZRj8jf4yx7ZMe8EK

Le CNN sono fondamentali per le attività di riconoscimento di immagini e video, in cui spesso è necessaria l'elaborazione in tempo reale. Possono essere ottimizzate per i sistemi embedded utilizzando tecniche come la quantizzazione e il "pruning" [potatura] per ridurre al minimo l'utilizzo della memoria e le richieste computazionali, consentendo funzionalità efficienti di rilevamento di oggetti e riconoscimento facciale in dispositivi con risorse computazionali limitate.

Esercizio 3.2: Convolutional Neural Networks (CNNs)

Abbiamo discusso del fatto che le CNN [Reti neurali convoluzionali] sono eccellenti nell'identificare le caratteristiche delle immagini, il che le rende ideali per attività come la classificazione degli oggetti. Ora, si potrà mettere in pratica questa conoscenza! Questo notebook Colab si concentra sulla creazione di una CNN per classificare le immagini dal set di dati CIFAR-10, che include oggetti come aeroplani, automobili e animali. Si impareranno le principali differenze tra CIFAR-10 e il set di dati MNIST che abbiamo esplorato in precedenza e come queste differenze influenzano la scelta del modello. Alla fine di questo notebook si avrà una conoscenza delle CNN per il riconoscimento delle immagini.

Open in Colab

3.2.4.3 Recurrent Neural Networks (RNN)

Le RNN [Reti Neurali Ricorrenti] sono adatte per l'analisi di dati sequenziali, come la previsione di serie temporali e l'elaborazione del linguaggio naturale. In questa architettura, le connessioni tra i nodi formano un grafo diretto lungo una sequenza temporale, consentendo il trasporto delle informazioni attraverso le sequenze tramite vettori di stato nascosti. Le varianti delle RNN includono le Long Short-Term Memory (LSTM) e le Gated Recurrent Units (GRU), progettate per catturare dipendenze più lunghe nei dati sequenziali. Queste reti possono essere utilizzate nei sistemi di riconoscimento vocale, nella manutenzione predittiva o nei dispositivi IoT in cui sono comuni i pattern di dati sequenziali. Le ottimizzazioni specifiche per le piattaforme embedded possono aiutare a gestirne i requisiti di elaborazione e memoria tipicamente elevati.

3.2.4.4 Generative Adversarial Network (GAN)

Le GAN [Reti Generative Avversarie] sono costituite da due reti, un generatore e un discriminatore, addestrate simultaneamente tramite l'addestramento adversarial [avversario] (Goodfellow et al. 2020). Il generatore produce dati che tentano di imitare la distribuzione di quelli reali, mentre il discriminatore distingue tra dati reali e dati generati. Le GAN sono ampiamente utilizzate nella generazione di immagini, nel trasferimento di stile e nell'aumento dei dati.

In contesti embedded, le reti GAN potrebbero essere utilizzate per l'aumento dei dati sul dispositivo per migliorare il training dei modelli direttamente sul dispositivo embedded, consentendo un apprendimento continuo e un adattamento ai nuovi dati senza la necessità di risorse di cloud computing.

3.2.4.5 Autoencoder

Gli autoencoder sono reti neurali per la compressione dei dati e la riduzione del rumore (Bank, Koenigstein, e Giryes 2023). Sono strutturati per codificare i dati di input in una rappresentazione a dimensione inferiore e quindi decodificarli nella loro forma originale. Varianti come gli Variational Autoencoders (VAE) [Autoencoder Variazionali] introducono livelli probabilistici che consentono proprietà generative, trovando applicazioni nella generazione di immagini e nel rilevamento di anomalie.

L'uso degli autoencoder può aiutare nella trasmissione e nell'archiviazione efficiente dei dati, migliorando le prestazioni complessive dei sistemi embedded con risorse di calcolo e di memoria limitate.

3.2.4.6 Transformer Network

Le "Transformer network" [reti di trasformatori] sono emerse come un'architettura potente, specialmente nell'elaborazione del linguaggio naturale (Vaswani et al. 2017). Queste reti utilizzano meccanismi di auto-attenzione per soppesare l'influenza di diverse parole di input su ogni parola di output, consentendo il calcolo parallelo e catturando pattern intricati nei

dati. Le reti di trasformatori hanno portato a risultati all'avanguardia in attività come la traduzione linguistica, la sintesi e la generazione di testo.
Queste reti possono essere ottimizzate per eseguire attività correlate alla lingua direttamente sul dispositivo. Ad esempio, i trasformatori possono essere utilizzati nei sistemi embedded per servizi di traduzione in tempo reale o interfacce assistite dalla voce, dove latenza ed efficienza computazionale sono cruciali. Tecniche come la distillazione del modello possono essere impiegate per distribuire queste reti su dispositivi embedded con risorse limitate.
Queste architetture servono a scopi specifici ed eccellono in diversi domini, offrendo un ricco toolkit per affrontare diversi problemi nei sistemi di intelligenza artificiale embedded. Comprendere le sfumature di queste architetture è fondamentale nella progettazione di modelli di deep learning efficaci ed efficienti per varie applicazioni.

3.2.5 ML Tradizionale vs Deep Learning

Il deep learning estende il machine learning tradizionale utilizzando reti neurali per discernere i pattern nei dati. Al contrario, il machine learning tradizionale si basa su un set di algoritmi consolidati come alberi decisionali, k-nearest neighbor e macchine a vettori di supporto, ma non coinvolge le reti neurali. Figura 3.8 fornisce un confronto visivo tra Machine Learning e Deep Learning, evidenziandone le principali differenze di approccio e capacità.

Figura 3.8: Confronto tra Machine Learning e Deep Learning. Fonte: Medium

Come mostrato nella figura, i modelli di deep learning possono elaborare dati grezzi direttamente ed estrarre automaticamente le feature rilevanti, mentre il machine learning tradizionale richiede spesso l'ingegneria manuale delle feature. La figura illustra anche come i modelli di deep learning possono gestire attività più complesse e set di dati più grandi rispetto ai tradizionali approcci di machine learning.
Per evidenziare ulteriormente le differenze, Tabella 3.1 fornisce un confronto più dettagliato delle caratteristiche contrastanti tra ML tradizionale e deep learning. Questa tabella integra la rappresentazione visiva in Figura 3.8 offrendo punti di confronto specifici tra vari aspetti di questi due approcci.

Tabella 3.1: Confronto tra machine learning tradizionale e deep learning.

Aspetto	ML tradizionale | Deep Learning
Requisiti dei dati	Da basso a moderato (efficiente con set di dati più piccoli) | Alto (richiede set di dati di grandi dimensioni per un apprendimento adeguato)

Complessità del modello	Moderata (adatta a problemi ben definiti)	Alta (rileva pattern intricati, adatta a compiti complessi)
Risorse di calcolo	Da basse a moderate (economiche, meno dispendiose in termini di risorse)	Alta (richiede una potenza di calcolo e risorse sostanziali)
Velocità di distribuzione	Veloce (cicli di training e distribuzione più rapidi)	Lento (tempi di training più lunghi, in particolare con con set di dati più grandi)
Interpretabilità	Alta (chiare intuizioni sui percorsi decisionali)	Bassa (strutture complesse a layer, natura “scatola nera”)
Manutenzione	Più facile (semplice da aggiornare e mantenere)	Complesso (richiede più sforzi nella manutenzione e negli aggiornamenti)

3.2.6 Scelta tra ML tradizionale e DL

3.2.6.1 Disponibilità e Volume dei Dati

Quantità di Dati: Gli algoritmi di machine learning tradizionali, come gli alberi decisionali o Naive Bayes, sono spesso più adatti quando la disponibilità dei dati è limitata. Offrono previsioni affidabili anche con set di dati più piccoli. Ciò è particolarmente vero nella diagnostica medica per la previsione delle malattie e nella segmentazione dei clienti nel marketing.

Diversità e Qualità dei Dati: Gli algoritmi di machine learning tradizionali spesso funzionano bene con dati strutturati (l’input del modello è un set di funzionalità, idealmente indipendenti l’una dall’altra) ma possono richiedere un notevole sforzo di pre-elaborazione (ad esempio, la “feature engineering” [progettazione delle funzionalità]). D’altro canto, il deep learning adotta l’approccio di eseguire automaticamente la progettazione delle funzionalità come parte dell’architettura del modello. Questo approccio consente la costruzione di modelli end-to-end in grado di mappare direttamente da dati di input non strutturati (come testo, audio e immagini) all’output desiderato senza fare affidamento su euristiche semplicistiche con efficacia limitata. Tuttavia, ciò si traduce in modelli più grandi che richiedono più dati e risorse computazionali. Nei dati rumorosi, la necessità di set di dati più grandi è ulteriormente enfatizzata quando si utilizza il Deep Learning.

3.2.6.2 Complessità del Problema

Granularità del Problema: I problemi che sono semplici o moderatamente complessi, che possono coinvolgere relazioni lineari o polinomiali tra variabili, spesso trovano una migliore aderenza ai metodi tradizionali di apprendimento automatico.

Rappresentazione Gerarchica delle Feature: I modelli di deep learning sono eccellenti in attività che richiedono una rappresentazione gerarchica delle feature caratteristiche, come il riconoscimento di immagini e voce. Tuttavia, non tutti i problemi richiedono questa complessità e gli algoritmi tradizionali di apprendimento automatico possono talvolta offrire soluzioni più semplici e ugualmente efficaci.

3.2.6.3 Risorse Hardware e Computazionali

Vincoli di Risorse: La disponibilità di risorse computazionali spesso influenza la scelta tra ML tradizionale e deep learning. Il primo è generalmente meno dispendioso in termini di risorse e quindi preferibile in ambienti con limitazioni hardware o vincoli di budget.

Scalabilità e Velocità: Gli algoritmi tradizionali di apprendimento automatico, come le Support Vector Machines (SVM) [macchine a vettori di supporto], spesso consentono tempi di training più rapidi e una scalabilità più semplice, il che è particolarmente vantaggioso nei progetti con tempistiche ristrette e volumi di dati in crescita.

3.2.6.4 Normativa di Conformità

La conformità normativa è fondamentale in vari settori, e richiede l'aderenza a linee guida e "best practice" come il General Data Protection Regulation (GDPR) [Regolamento generale sulla protezione dei dati] nell'UE. I modelli ML tradizionali, grazie alla loro intrinseca interpretabilità, spesso si allineano meglio a queste normative, soprattutto in settori come la finanza e l'assistenza sanitaria.

3.2.6.5 Interpretabilità

Comprendere il processo decisionale è più facile con le tecniche tradizionali di apprendimento automatico rispetto ai modelli di deep learning, che funzionano come "scatole nere", rendendo difficile tracciare i percorsi decisionali.

3.2.7 Fare una Scelta Informata

Considerati i vincoli dei sistemi di intelligenza artificiale embedded, comprendere le differenze tra le tecniche di ML tradizionali e il deep learning diventa essenziale. Entrambe le strade offrono vantaggi unici e le loro caratteristiche distintive spesso determinano la scelta dell'una rispetto all'altra in diversi scenari.

Nonostante ciò, il deep learning ha costantemente superato i metodi tradizionali di apprendimento automatico in diverse aree chiave grazie all'abbondanza di dati, ai progressi computazionali e alla comprovata efficacia in attività complesse. Ecco alcuni motivi specifici per cui ci concentriamo sul deep learning:

1. **Prestazioni Superiori in Attività Complesse:** I modelli di deep learning, in particolare le reti neurali profonde, eccellono in attività in cui le relazioni tra i punti dati sono incredibilmente intricate. Attività come il riconoscimento di immagini e parlato, la traduzione linguistica e la riproduzione di giochi complessi come Go e Scacchi hanno visto progressi significativi principalmente attraverso algoritmi di deep learning.
2. **Gestione Efficiente dei Dati non Strutturati:** A differenza dei metodi tradizionali di apprendimento automatico, il deep learning può elaborare in modo più efficace i dati non strutturati. Ciò è fondamentale nel panorama dei dati odierno, in cui la stragrande maggioranza dei dati, come testo, immagini e video, non è strutturata.
3. **Sfruttamento dei Big Data:** Con la disponibilità dei Big Data, i modelli di deep learning possono apprendere e migliorare continuamente. Questi modelli eccellono nell'utilizzare grandi set di dati per migliorare la loro accuratezza predittiva, un limite degli approcci tradizionali di machine-learning.
4. **Progressi Hardware e Calcolo Parallelo:** L'avvento di potenti GPU e la disponibilità di piattaforme di cloud computing hanno consentito il rapido training di modelli di deep learning. Questi progressi hanno affrontato una delle sfide significative del deep learning: la necessità di risorse computazionali sostanziali.
5. **Adattabilità Dinamica e Apprendimento Continuo:** I modelli di deep learning possono adattarsi dinamicamente a nuove informazioni o dati. Possono essere addestrati per generalizzare il loro apprendimento a nuovi dati mai visti, cruciali in campi in rapida evoluzione come la guida autonoma o la traduzione linguistica in tempo reale.

Sebbene il deep learning abbia guadagnato una notevole popolarità, è essenziale comprendere che il machine learning tradizionale è ancora rilevante. Man mano che ci addentriamo nei meandri del deep learning, evidenzieremo anche le situazioni in cui i metodi

tradizionali di machine learning potrebbero essere più appropriati, grazie alla loro semplicità, efficienza e interpretabilità. Concentrandoci in questo testo sul deep learning, intendiamo fornire ai lettori le conoscenze e gli strumenti per affrontare problemi moderni e complessi in vari ambiti, fornendo al contempo approfondimenti sui vantaggi comparativi e sugli scenari applicativi appropriati per il deep learning e le tecniche tradizionali di machine learning.

3.3 Conclusione

Il deep learning è diventato un potente set di tecniche per affrontare le complesse sfide del riconoscimento di pattern e della previsione. Iniziando con una panoramica, abbiamo delineato i concetti e i principi fondamentali che governano il deep learning, gettando le basi per studi più avanzati.

Al centro del deep learning, abbiamo esplorato le idee di base delle reti neurali, potenti modelli computazionali ispirati alla struttura neuronale interconnessa del cervello umano. Questa esplorazione ci ha permesso di apprezzare le capacità e il potenziale delle reti neurali nella creazione di algoritmi sofisticati in grado di apprendere e adattarsi dai dati.

Comprendere il ruolo delle librerie e dei framework è stata una parte fondamentale della nostra discussione. Abbiamo offerto approfondimenti sugli strumenti che possono facilitare lo sviluppo e l'implementazione di modelli di deep learning. Queste risorse semplificano l'implementazione delle reti neurali e aprono strade all'innovazione e all'ottimizzazione.

Successivamente, abbiamo affrontato le sfide che si potrebbero incontrare quando si racchiudono algoritmi di deep learning nei sistemi embedded, fornendo una prospettiva critica sulle complessità e sulle considerazioni relative all'introduzione dell'intelligenza artificiale nei dispositivi edge.

Inoltre, abbiamo esaminato i limiti del deep learning. Attraverso le discussioni, abbiamo svelato le sfide affrontate nelle applicazioni del deep learning e delineato scenari in cui l'apprendimento automatico tradizionale potrebbe superare il deep learning. Queste sezioni sono fondamentali per promuovere una visione equilibrata delle capacità e dei limiti del deep learning.

In questo "Avviamento", abbiamo fornito le conoscenze per fare scelte informate tra l'implementazione dell'apprendimento automatico tradizionale o delle tecniche di deep learning, a seconda delle esigenze e dei vincoli unici di un problema specifico.

Concludendo questo capitolo, ci auguriamo che sia stato acquisito il "linguaggio" di base del deep learning e si sia pronti ad approfondire i capitoli successivi con una solida comprensione e una prospettiva critica. Il viaggio che è pieno di entusiasmanti opportunità e sfide nel racchiudere l'intelligenza artificiale nei sistemi.

3.4 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Stiamo lavorando costantemente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

ⓘ Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Past, Present, and Future of ML.
- Thinking About Loss.
- Minimizing Loss.
- First Neural Network.
- Understanding Neurons.
- Intro to CLassification.
- Training, Validation, and Test Data.
- Intro to Convolutions.

Video

- Video 3.3
- Video 3.1
- Video 3.2

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 3.1
- Esercizio 3.2

Laboratori

- *Prossimamente*.

4. Workflow dell'IA

DALL·E 3 Prompt: Creare un'illustrazione rettangolare di un diagramma di flusso stilizzato che rappresenti il flusso di lavoro/pipeline dell'IA. Da sinistra a destra, rappresenta le fasi come segue: "Raccolta dati" con un'icona di database, "Preelaborazione dati" con un'icona di filtro, "Progettazione modello" con un'icona di un cervello, "Addestramento" con un'icona di un peso, "Valutazione" con un segno di spunta e "Distribuzione" con un razzo. Collegare ogni fase con delle frecce per guidare l'osservatore orizzontalmente attraverso i processi dell'IA, sottolineando la natura sequenziale e interconnessa di questi passaggi.

Il flusso di lavoro del ML è un approccio strutturato che guida professionisti e ricercatori attraverso lo sviluppo, l'implementazione e la manutenzione di modelli di ML. Questo flusso di lavoro è generalmente suddiviso in diverse fasi cruciali, ciascuna delle quali contribuisce allo sviluppo efficace di sistemi intelligenti.
In questo capitolo esploreremo il flusso di lavoro del machine learning, ponendo le basi per i capitoli successivi che approfondiranno le specifiche. Questo capitolo si concentra solo sulla presentazione di una panoramica di alto livello dei passaggi coinvolti nel flusso di lavoro ML.

Obiettivi dell'Apprendimento

- Comprendere il flusso di lavoro ML e acquisire informazioni sull'approccio strutturato e sulle fasi di sviluppo, distribuzione e manutenzione dei modelli di apprendimento automatico.
- Scoprire le sfide e le distinzioni uniche tra i flussi di lavoro per il machine learning tradizionale e l'intelligenza artificiale embedded.
- Apprezzare i ruoli nei progetti di ML e comprenderne le responsabilità e il significato.
- Comprendere l'importanza, le applicazioni e le considerazioni per l'implementazione dei modelli ML in ambienti con risorse limitate.
- Acquisire consapevolezza degli aspetti etici e legali che devono essere considerati e rispettati nei progetti ML e intelligenza artificiale embedded.
- Stabilire una comprensione di base dei flussi di lavoro e dei ruoli di ML per essere ben preparati per un'esplorazione più approfondita nei capitoli seguenti.

4.1 Panoramica

Figura 4.1: Metodologia di progettazione multi-step per lo sviluppo di un modello di machine learning. Comunemente denominato ciclo di vita del machine learning

Figura 4.1 illustra il flusso di lavoro sistematico necessario per sviluppare un modello di machine learning di successo. Questo processo end-to-end, comunemente denominato ciclo di vita del machine learning, consente di creare, distribuire e gestire i modelli in modo efficace. Solitamente comporta i seguenti passaggi chiave:

1. **Definizione del Problema** - Si inizia articolando chiaramente il problema specifico da risolvere. Questo si concentra sui problemi durante la raccolta dati e la creazione del modello.
2. **Raccolta e Preparazione dei Dati:** Raccogliere dati di training pertinenti e di alta qualità che catturino tutti gli aspetti del problema. Pulire e pre-elaborare i dati per prepararli alla modellazione.
3. **Selezione e Training del Modello:** Scegliere un algoritmo di apprendimento automatico adatto al tipo di problema e ai dati. Considerare i pro e i contro dei diversi approcci. Inserire i dati preparati nel modello per addestrarlo. Il tempo di addestramento varia in base alle dimensioni dei dati e alla complessità del modello.
4. **Valutazione del Modello:** Testare il modello addestrato su nuovi dati non ancora esaminati per misurarne l'accuratezza predittiva. Identificare eventuali limitazioni.
5. **Distribuzione del Modello:** Integrare il modello convalidato in applicazioni o sistemi per avviarne l'operatività.
6. **Monitoraggio e Manutenzione:** Tenere traccia delle prestazioni del modello in produzione. Ri-addestrare periodicamente su nuovi dati per mantenerli aggiornati.

Seguire questo flusso di lavoro ML strutturato ci guida attraverso le fasi chiave dello sviluppo. Garantisce di creare modelli efficaci e robusti pronti per la distribuzione nel mondo reale, con conseguenti modelli di qualità superiore che risolvono le varie esigenze.
Il flusso di lavoro ML è iterativo, richiede un monitoraggio continuo e potenziali aggiustamenti. Ulteriori considerazioni includono:

- **Controllo della Versione:** Tenere traccia delle modifiche al codice e ai dati per riprodurre i risultati e ripristinare le versioni precedenti se necessario.

- **Documentazione:** Mantenere una documentazione dettagliata per la comprensione e la riproduzione del flusso di lavoro.
- **Test:** Testare rigorosamente il flusso di lavoro per garantirne la funzionalità.
- **Sicurezza:** Proteggere il flusso di lavoro e i dati quando si distribuiscono modelli in contesti di produzione.

4.2 IA Tradizionale o Embedded

Il flusso di lavoro ML è una guida universale applicabile su diverse piattaforme, tra cui soluzioni basate su cloud, edge computing e TinyML. Tuttavia, il flusso di lavoro per l'IA Embedded introduce complessità e sfide uniche, rendendolo un dominio accattivante e aprendo la strada a innovazioni straordinarie. Figura 4.2 illustra le differenze tra Machine Learning e Deep Learning.

Figura 4.2: Confronto tra Machine Learning tradizionale e Deep Learning. Fonte: BBN Times

Figura 4.3 illustra gli utilizzi dell'intelligenza artificiale embedded in vari settori.

Figura 4.3: Applicazioni di IA Embedded. Fonte: Rinf.tech

4.2.1 Ottimizzazione delle Risorse

- **Flusso di Lavoro ML Tradizionale:** Questo workflow dà priorità all'accuratezza e alle prestazioni del modello, spesso sfruttando abbondanti risorse di calcolo in ambienti cloud o data center.
- **Flusso di Lavoro IA Embedded:** Dati i vincoli di risorse dei sistemi embedded, questo flusso di lavoro richiede un'attenta pianificazione per ottimizzare le dimensioni del modello e le richieste di calcolo. Tecniche come la quantizzazione e il pruning [potatura] del modello sono fondamentali.

4.2.2 Elaborazione in Real-time

- **Flusso di Lavoro ML Tradizionale:** Meno enfasi sull'elaborazione in tempo reale, spesso basata sull'elaborazione di dati in batch.
- **Flusso di Lavoro IA Embedded:** Dà priorità all'elaborazione dei dati in tempo reale, rendendo essenziali bassa latenza ed esecuzione rapida, soprattutto in applicazioni come veicoli autonomi e automazione industriale.

4.2.3 Gestione dei Dati e Privacy

- **Flusso di Lavoro ML Tradizionale:** Elabora i dati in posizioni centralizzate, spesso richiedendo un ampio trasferimento di dati e concentrandosi sulla sicurezza dei dati durante il transito e l'archiviazione.
- **Flusso di Lavoro IA Embedded:** Questo workflow sfrutta l'edge computing per elaborare i dati più vicino alla fonte, riducendo la trasmissione dei dati e migliorando la privacy tramite la localizzazione dei dati.

4.2.4 Integrazione Hardware-Software

- **Flusso di Lavoro ML Tradizionale:** In genere funziona su hardware generico, con sviluppo di software indipendente.
- **Flusso di Lavoro IA Embedded:** Questo flusso di lavoro prevede un approccio più integrato allo sviluppo hardware e software, spesso incorporando chip personalizzati o acceleratori hardware per ottenere prestazioni ottimali.

4.3 Ruoli e responsabilità

La creazione di una soluzione ML, in particolare per l'intelligenza artificiale embedded, è uno sforzo multidisciplinare che coinvolge vari specialisti. A differenza dello sviluppo software tradizionale, la creazione di una soluzione ML richiede un approccio multidisciplinare a causa della natura sperimentale dello sviluppo del modello e dei requisiti ad alta intensità di risorse per il training e l'implementazione di questi modelli.
C'è una forte necessità di ruoli incentrati sui dati per il successo delle pipeline di apprendimento automatico. Gli scienziati dei dati e gli ingegneri dei dati gestiscono la raccolta dei dati, creano pipeline di dati e ne garantiscono la qualità. Poiché la natura dei modelli di apprendimento automatico dipende dai dati che consumano, i modelli sono unici e variano a seconda delle diverse applicazioni, il che richiede un'ampia sperimentazione. I ricercatori e gli ingegneri di apprendimento automatico guidano questa fase sperimentale attraverso test continui, convalida e iterazione per ottenere prestazioni ottimali.
La fase di implementazione richiede spesso hardware e infrastrutture specializzati, poiché i modelli di machine learning possono essere ad alta intensità di risorse, richiedendo un'elevata potenza di calcolo e una gestione efficiente delle risorse. Ciò richiede la collaborazione con gli ingegneri hardware per garantire che l'infrastruttura possa supportare le esigenze computazionali di training e inferenza del modello.

Poiché i modelli prendono decisioni che possono avere un impatto sugli individui e sulla società, gli aspetti etici e legali dell'apprendimento automatico stanno diventando sempre più importanti. Sono necessari esperti di etica e consulenti legali per garantire la conformità agli standard etici e alle normative legali.
Comprendere i vari ruoli coinvolti in un progetto ML è fondamentale per il suo completamento con successo. Tabella 4.1 fornisce una panoramica generale di questi ruoli tipici, anche se è importante notare che i confini tra loro a volte possono essere confusi. Esaminiamo questa ripartizione:

Tabella 4.1: Ruoli e responsabilità delle persone coinvolte in Operazioni di ML.

Ruolo	Responsabilità
Project Manager	Supervisiona il progetto, assicurando che le tempistiche e le milestone siano rispettate.
Esperti di Dominio	Offrono approfondimenti specifici del dominio per definire i requisiti del progetto.
Data Scientist	Specializzati nell'analisi dei dati e nello sviluppo di modelli.
Ingegneri di Apprendimento Automatico	Concentrati sullo sviluppo e l'implementazione del modello.
Data Scientist	Specializzati nell'analisi dei dati e nello sviluppo di modelli.
Embedded Systems Engineer	Integra modelli ML in sistemi embedded.
Software Developer	Sviluppa componenti software per l'integrazione del sistema IA.
Hardware Engineer	Progetta e ottimizza l'hardware per il sistema AI embedded.
UI/UX Designer	Concentrato sulla progettazione incentrata sull'utente.
QA Engineer	Assicura che il sistema soddisfi gli standard di qualità.
Eticisti e Consulenti Legali	Consulenti sulla conformità etica e legale.
Personale Operativo e di Manutenzione	Monitora e mantiene il sistema distribuito.
Specialisti della sicurezza	Garantiscono la sicurezza del sistema.

Man mano che procederemo nei prossimi capitoli, esploreremo l'essenza e le competenze di ciascun ruolo e favoriremo una comprensione più profonda delle complessità coinvolte nei progetti di intelligenza artificiale. Questa visione olistica facilita una collaborazione senza soluzione di continuità e alimenta un ambiente maturo per innovazione e scoperte.

4.4 Conclusione

Questo capitolo ha gettato le basi per comprendere il flusso di lavoro dell'apprendimento automatico, un approccio strutturato fondamentale per lo sviluppo, l'implementazione e la manutenzione dei modelli ML. Abbiamo esplorato le sfide uniche affrontate nei flussi di lavoro ML, dove l'ottimizzazione delle risorse, l'elaborazione in tempo reale, la gestione dei dati e l'integrazione hardware-software sono fondamentali. Queste distinzioni sottolineano l'importanza di adattare i flussi di lavoro per soddisfare le esigenze specifiche dell'ambiente applicativo.
Inoltre, abbiamo sottolineato l'importanza della collaborazione multidisciplinare nei progetti ML. Esaminando i diversi ruoli coinvolti, dai data scientist agli ingegneri del software,

abbiamo ottenuto una panoramica del lavoro di squadra necessario per affrontare la natura sperimentale e ad alta intensità di risorse dello sviluppo ML. Questa comprensione è fondamentale per promuovere una comunicazione e una collaborazione efficaci tra diversi settori di competenza.
Mentre passiamo a discussioni più dettagliate nei capitoli successivi, questa panoramica di alto livello ci fornisce una prospettiva olistica sul flusso di lavoro ML e sui vari ruoli coinvolti. Queste basi si riveleranno importanti quando approfondiremo aspetti specifici dell'apprendimento automatico, che ci consentiranno di contestualizzare concetti avanzati nel quadro più ampio dello sviluppo e dell'implementazione del machine learning.

4.5 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- ML Workflow.
- ML Lifecycle.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- *Prossimamente*.

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

5. Data Engineering

DALL·E 3 Prompt: Creare un'illustrazione rettangolare che visualizzi il concetto di data engineering. Includere elementi quali fonti di dati grezzi, pipeline di elaborazione dati, sistemi di archiviazione e set di dati raffinati. Mostrare come i dati grezzi vengono trasformati tramite pulizia, elaborazione e archiviazione per diventare informazioni preziose che possono essere analizzate e utilizzate per il processo decisionale.

I dati sono la linfa vitale dei sistemi di intelligenza artificiale. Senza buoni dati, anche gli algoritmi di apprendimento automatico più avanzati non avranno successo. Tuttavia, i modelli TinyML operano su dispositivi con potenza di elaborazione e memoria limitate. Questa sezione esplora le complessità della creazione di set di dati di alta qualità per alimentare i nostri modelli di intelligenza artificiale. Il data engineering implica la raccolta, l'archiviazione, l'elaborazione e la gestione dei dati per addestrare modelli di apprendimento automatico.

Obiettivi dell'Apprendimento

- Comprendere l'importanza di definire chiaramente il problema e gli obiettivi quando si intraprende un progetto di ML.
- Riconoscere varie tecniche di sourcing dei dati, come web scraping, crowdsourcing e generazione di dati sintetici, insieme ai loro vantaggi e limiti.
- Comprendere la necessità di un'etichettatura dei dati ponderata, utilizzando approcci manuali o assistiti dall'intelligenza artificiale, per creare set di dati di training di alta qualità.
- Imparare brevemente diversi metodi per archiviare e gestire i dati, come database, data warehouse e data lake.
- Comprendere il ruolo della trasparenza attraverso la documentazione di metadati e set di dati e il monitoraggio della provenienza dei dati per facilitare l'etica, l'audit e la riproducibilità.
- Comprendere come i protocolli di licenza regolano l'accesso e l'utilizzo dei dati legali, richiedendo un'attenta conformità.
- Riconoscere le principali sfide nell'ingegneria dei dati, tra cui rischi per la privacy, lacune di rappresentazione, restrizioni legali sull'accesso ai dati e bilanciamento delle priorità concorrenti.

5.1 Introduzione

Si immagini un mondo in cui l'intelligenza artificiale può diagnosticare malattie con una precisione senza precedenti, ma solo se i dati utilizzati per addestrarla sono imparziali e affidabili. È qui che entra in gioco il "data engineering" [ingegneria dei dati]. Sebbene oltre il 90% dei dati mondiali sia stato creato negli ultimi due decenni, questa enorme quantità di informazioni è utile solo per creare modelli di intelligenza artificiale efficaci con un'elaborazione e una preparazione adeguate. L'ingegneria dei dati colma questa lacuna trasformando i dati grezzi in un formato di alta qualità che alimenta l'innovazione dell'intelligenza artificiale. Nel mondo odierno basato sui dati, proteggere la privacy degli utenti è fondamentale. Che siano obbligatorie per legge o guidate dalle preoccupazioni degli utenti, le tecniche di anonimizzazione come la privacy differenziale e l'aggregazione sono fondamentali per mitigare i rischi per la privacy. Tuttavia, un'implementazione attenta è fondamentale per garantire che questi metodi non compromettano l'utilità dei dati. I creatori di set di dati affrontano complesse sfide di privacy e rappresentazione quando creano dati di addestramento di alta qualità, in particolare per domini sensibili come l'assistenza sanitaria. Dal punto di vista legale, i creatori potrebbero dover rimuovere identificatori diretti come nomi ed età. Anche senza obblighi legali, la rimozione di tali informazioni può aiutare a creare fiducia negli utenti. Tuttavia, un'eccessiva anonimizzazione può compromettere l'utilità del set di dati. Tecniche come la privacy differenziale[1], l'aggregazione e la riduzione dei dettagli forniscono alternative per bilanciare privacy e utilità, ma hanno degli svantaggi. I creatori devono trovare un equilibrio ponderato in base al caso d'uso.

Sebbene la privacy sia fondamentale, garantire modelli di intelligenza artificiale equi e solidi richiede di affrontare le lacune (gap) della rappresentazione nei dati. È fondamentale ma non sufficiente garantire la diversità tra variabili individuali come genere, razza e accento. Queste combinazioni, a volte chiamate lacune (gap) di ordine superiore, possono influire in modo significativo sulle prestazioni del modello. Ad esempio, un set di dati medico potrebbe avere dati bilanciati su genere, età e diagnosi individualmente, ma non ha abbastanza casi per catturare donne anziane con una condizione specifica. Tali higher-order gaps [lacune di ordine superiore] non sono immediatamente evidenti, ma possono influire in modo critico sulle prestazioni del modello.

La creazione di dati di training utili ed etici richiede una considerazione globale dei rischi per la privacy e delle lacune di rappresentazione. Le soluzioni perfette elusive necessitano di pratiche di ingegneria dei dati coscienziose come l'anonimizzazione, l'aggregazione, il sotto-campionamento di gruppi sovrarappresentati e la generazione di dati sintetizzati per bilanciare esigenze contrastanti. Ciò facilita modelli che sono sia accurati che socialmente responsabili. La collaborazione interfunzionale e i controlli esterni possono anche rafforzare i dati di training. Le sfide sono molteplici ma superabili con uno sforzo ponderato.

Iniziamo discutendo della raccolta dati: Dove reperiamo i dati e come li raccogliamo? Le opzioni spaziano dall'estrazione di dati dal web, all'accesso alle API e all'utilizzo di sensori e dispositivi IoT, fino alla conduzione di sondaggi e alla raccolta di input dagli utenti. Questi metodi riflettono pratiche del mondo reale. Successivamente, approfondiremo l'etichettatura dei dati, tenendo conto anche del coinvolgimento umano. Discuteremo i compromessi e le limitazioni dell'etichettatura umana ed esploreremo i metodi emergenti per l'etichettatura automatizzata. Successivamente, affronteremo la pulizia e la preelaborazione dei dati, un passaggio cruciale ma spesso sottovalutato nella preparazione dei dati grezzi per l'addestramento del modello di intelligenza artificiale. Segue l'aumento dei dati, una strategia per migliorare set di dati limitati generando campioni sintetici. Ciò è particolarmente pertinente per i sistemi embedded, poiché molti casi d'uso necessitano di ampi repository di dati prontamente disponibili per la cura

[https://it.wikipedia.org/wiki/Data_curation]. La generazione di dati sintetici emerge come un'alternativa praticabile con vantaggi e svantaggi. Parleremo anche del versioning del dataset, sottolineando l'importanza di tracciare le modifiche dei dati nel tempo. I dati sono in continua evoluzione; quindi, è fondamentale ideare strategie per gestire e archiviare dataset espansivi. Alla fine di questa sezione, si avrà una comprensione completa dell'intera pipeline di dati, dalla raccolta all'archiviazione, essenziale per rendere operativi i sistemi di intelligenza artificiale. Intraprendiamo questo viaggio!

5.2 Definizione del Problema

In molti domini di machine learning, algoritmi sofisticati sono al centro dell'attenzione, mentre l'importanza fondamentale della qualità dei dati viene spesso trascurata. Questa negligenza dà origine a "Data Cascades" di Sambasivan et al. (2021a), eventi in cui le lacune nella qualità dei dati si sommano, portando a conseguenze negative a valle come previsioni errate, cessazioni di progetti e persino potenziali danni alle comunità.
Figura 5.1 illustra queste potenziali insidie nei dati in ogni fase e come influenzano l'intero processo lungo la linea. L'influenza degli errori nella raccolta dei dati è particolarmente pronunciata. Come illustrato nella figura, qualsiasi lacuna in questa fase iniziale diventerà evidente nelle fasi successive (nella valutazione e nell'implementazione del modello) e potrebbe portare a conseguenze costose, come l'abbandono dell'intero modello e il riavvio da zero. Pertanto, investire in tecniche di ingegneria dei dati fin dall'inizio ci aiuterà a rilevare gli errori in anticipo, mitigando gli effetti a cascata illustrati nella figura.

Figura 5.1: Data cascades: costi composti. Fonte: Sambasivan et al. (2021a).

Nonostante molti professionisti del ML riconoscano l'importanza dei dati, altri segnalano di dover affrontare queste "cascate". Ciò evidenzia un problema sistemico: mentre il fascino dello sviluppo di modelli avanzati rimane, i dati spesso devono essere maggiormente apprezzati.
Keyword Spotting (KWS) fornisce un esempio eccellente di TinyML in azione, come illustrato in Figura 5.2. Questa tecnologia è fondamentale per le interfacce abilitate alla voce su dispositivi endpoint come gli smartphone. In genere funzionando come motori di wake-word leggeri, i sistemi KWS sono costantemente attivi, in ascolto di una frase specifica per attivare ulteriori azioni. Come illustrato nella figura, quando diciamo "OK, Google" o "Alexa", questo avvia un processo su un microcontrollore embedded nel dispositivo. Nonostante le loro risorse limitate, questi microcontrollori svolgono un ruolo importante nel consentire interazioni vocali senza interruzioni con i dispositivi, spesso operando in ambienti con elevato rumore ambientale. L'unicità della wake-word, come mostrato nella figura, aiuta a ridurre al minimo i falsi positivi, assicurando che il sistema non venga attivato inavvertitamente.

Figura 5.2: Esempio di individuazione delle "Keyword Spotting": interazione con Alexa. Fonte: Amazon.

È importante comprendere che queste tecnologie di individuazione delle "parole chiave" non sono isolate; si integrano perfettamente in sistemi più grandi, elaborando segnali in modo continuo e gestendo al contempo un basso consumo energetico. Questi sistemi vanno oltre il semplice riconoscimento delle parole chiave, evolvendosi per facilitare diversi rilevamenti di suoni, come la rottura di un vetro. Questa evoluzione è orientata alla creazione di dispositivi intelligenti in grado di comprendere e rispondere ai comandi vocali, annunciando un futuro in cui anche gli elettrodomestici possono essere controllati tramite interazioni vocali.
Creare un modello KWS affidabile è un compito complesso. Richiede una profonda comprensione dello scenario di distribuzione, che comprenda dove e come funzioneranno questi dispositivi. Ad esempio, l'efficacia di un modello KWS non riguarda solo il riconoscimento di una parola; riguarda la sua distinzione tra vari accenti e rumori di sottofondo, che si tratti di un bar affollato o del suono stridulo di una televisione in un soggiorno o in una cucina dove questi dispositivi sono comunemente presenti. Riguarda la garanzia che un sussurrato "Alexa" nel cuore della notte o un urlato "OK Google" in un mercato rumoroso vengano riconosciuti con la stessa precisione.
Inoltre, molti degli attuali assistenti vocali KWS supportano un numero limitato di lingue, lasciando una parte sostanziale della diversità linguistica mondiale non rappresentata. Questa limitazione è in parte dovuta alla difficoltà di raccogliere e monetizzare i dati per le lingue parlate da popolazioni più piccole. La distribuzione "long-tail" [https://it.wikipedia.org/wiki/Coda_lunga] delle lingue implica che molte lingue hanno dati limitati, rendendo difficile lo sviluppo di tecnologie di supporto.
Questo livello di accuratezza e robustezza dipende dalla disponibilità e dalla qualità dei dati, dalla capacità di etichettare correttamente i dati e dalla trasparenza dei dati per l'utente finale prima che vengano utilizzati per addestrare il modello. Tuttavia, tutto inizia con una chiara comprensione della dichiarazione o definizione del problema.
In genere, in ML, la definizione del problema ha alcuni passaggi chiave:

1. Identificare chiaramente la definizione del problema
2. Definire obiettivi chiari
3. Stabilire un benchmark [riferimento] di successo

4. Comprendere l'impegno/l'uso dell'utente finale
5. Comprendere i vincoli e le limitazioni dell'implementazione
6. Seguito infine dalla raccolta dati.

Una solida base di progetto è essenziale per la sua traiettoria e il suo successo finale. Al centro di questa base c'è innanzitutto l'identificazione di un problema chiaro, come garantire che i comandi vocali nei sistemi di assistenza vocale siano riconosciuti in modo coerente in diversi ambienti. Obiettivi chiari, come la creazione di set di dati rappresentativi per scenari diversi, forniscono una direzione unificata. I benchmark, come l'accuratezza del sistema nel rilevamento delle parole chiave, offrono risultati misurabili per valutare i progressi. Il coinvolgimento delle parti interessate, dagli utenti finali agli investitori, fornisce informazioni preziose e garantisce l'allineamento con le esigenze del mercato. Inoltre, quando si esplorano ambiti come l'assistenza vocale, è importante comprendere i limiti della piattaforma. I sistemi embedded, come i microcontrollori, sono dotati di limitazioni intrinseche di potenza di elaborazione, memoria ed efficienza energetica. Riconoscere queste limitazioni garantisce che le funzionalità, come il rilevamento delle parole chiave, siano personalizzate per funzionare in modo ottimale, bilanciando le prestazioni col risparmio delle risorse.

In questo contesto, usando KWS come esempio, possiamo suddividere ciascuno dei passaggi come segue:

1. **Identificazione del Problema:** In sostanza, KWS rileva parole chiave specifiche tra suoni ambientali e altre parole pronunciate. Il problema principale è progettare un sistema in grado di riconoscere queste parole chiave con elevata accuratezza, bassa latenza e minimi falsi positivi o negativi, soprattutto se distribuito su dispositivi con risorse di elaborazione limitate.
2. **Impostazione di Obiettivi Chiari:** Gli obiettivi per un sistema KWS potrebbero includere:
    - Raggiungimento di un tasso di accuratezza specifico (ad esempio, accuratezza del 98% nel rilevamento delle parole chiave).
    - Garanzia di bassa latenza (ad esempio, rilevamento delle parole chiave e risposta entro 200 millisecondi).
    - Riduzione al minimo del consumo di energia per estendere la durata della batteria sui dispositivi embedded.
    - Garanzia che le dimensioni del modello siano ottimizzate per la memoria disponibile sul dispositivo.
3. **Benchmark per il successo:** Stabilire metriche chiare per misurare il successo del sistema KWS. Questo potrebbe includere:
    - Tasso di Veri Positivi: La percentuale di parole chiave identificate correttamente.
    - Tasso di Falsi Positivi: La percentuale di parole chiave non identificate erroneamente come parole chiave.
    - Tempo di Risposta: Il tempo impiegato dall'enunciazione della parola chiave alla risposta del sistema.
    - Consumo Energetico: Potenza media utilizzata durante il rilevamento della parola chiave.
4. **Coinvolgimento e Comprensione delle Parti Interessate::** Coinvolgere le parti interessate, tra cui produttori di dispositivi, sviluppatori di hardware e software e utenti finali. Comprendere le loro esigenze, capacità e vincoli. Ad esempio:
    - I produttori di dispositivi potrebbero dare priorità al basso consumo energetico.

    - Gli sviluppatori di software potrebbero enfatizzare la facilità di integrazione.
    - Gli utenti finali darebbero priorità all'accuratezza e alla reattività.
5. **Comprensione dei Vincoli e delle Limitazioni dei Sistemi Embedded:** I dispositivi embedded presentano una serie di problematiche:
    - Limiti della Memoria: I modelli KWS devono essere leggeri per adattarsi ai vincoli di memoria dei dispositivi embedded. In genere, i modelli KWS devono essere piccoli quanto 16 KB per adattarsi alla "isola always-on" [porzione sempre attiva] del SoC. Inoltre, questa è solo la dimensione del modello. Anche il codice applicativo aggiuntivo per la pre-elaborazione potrebbe dover rientrare nei vincoli di memoria.
    - Potenza di Elaborazione: Le capacità di calcolo dei dispositivi embedded sono limitate (alcune centinaia di MHz di velocità di clock), quindi il modello KWS deve essere ottimizzato per l'efficienza.
    - Consumo Energetico: Poiché molti dispositivi embedded sono alimentati a batteria, il sistema KWS deve essere efficiente dal punto di vista energetico.
    - Vincoli Ambientali: I dispositivi potrebbero essere distribuiti in vari ambienti, dalle silenziose camere da letto agli ambienti industriali rumorosi. Il sistema KWS deve essere sufficientemente robusto per funzionare efficacemente in questi scenari.
6. **Raccolta e Analisi dei Dati:** Per un sistema KWS, la qualità e la diversità dei dati sono fondamentali. Le considerazioni potrebbero includere:
    - Varietà di Accenti: Raccogliere dati da parlanti con accenti diversi per garantire un riconoscimento ad ampio raggio.
    - Rumori di Sottofondo: Includere campioni di dati con diversi rumori ambientali per addestrare il modello per scenari del mondo reale.
    - Variazioni delle Parole Chiave: Le persone potrebbero pronunciare le parole chiave in modo diverso o avere leggere variazioni nella parola di attivazione stessa. Assicurarsi che il set di dati catturi queste sfumature.
7. **Feedback e Perfezionamento Iterativo:** Una volta sviluppato un prototipo di sistema KWS, è fondamentale testarlo in scenari del mondo reale, raccogliere feedback e perfezionare iterativamente il modello. Ciò garantisce che il sistema rimanga allineato con il problema e gli obiettivi definiti. Ciò è importante perché gli scenari di distribuzione cambiano nel tempo man mano che le cose si evolvono.

Esercizio 5.1: Keyword Spotting con TensorFlow Lite Micro

Esplorare una guida pratica per la creazione e l'implementazione di sistemi Keyword Spotting utilizzando TensorFlow Lite Micro. Seguire i passaggi dalla raccolta dati all'addestramento del modello e all'implementazione nei microcontrollori. Imparare a creare modelli KWS efficienti che riconoscono parole chiave specifiche in mezzo al rumore di fondo. Perfetto per chi è interessato all'apprendimento automatico sui sistemi embedded. Sbloccare il potenziale dei dispositivi "voice-enabled" con TensorFlow Lite Micro!

Open in Colab

Il capitolo corrente sottolinea il ruolo essenziale della qualità dei dati nell'apprendimento automatico, utilizzando come esempio i sistemi Keyword Spotting. Descrive i passaggi chiave, dalla definizione del problema al coinvolgimento delle parti interessate, sottolineando il feedback iterativo. Il prossimo capitolo approfondirà la gestione della qualità

dei dati, discutendone le conseguenze e le tendenze future, concentrandosi sull'importanza di dati diversificati e di alta qualità nello sviluppo di sistemi di intelligenza artificiale, affrontando considerazioni etiche e metodi di reperimento dei dati.

5.3 Ricerca dei Dati.

La qualità e la diversità dei dati raccolti sono importanti per sviluppare sistemi di intelligenza artificiale accurati e robusti. Il reperimento di dati di training di alta qualità richiede un'attenta considerazione degli obiettivi, delle risorse e delle implicazioni etiche. I dati possono essere ottenuti da varie fonti a seconda delle esigenze del progetto:

5.3.1 Dataset preesistenti

Piattaforme come Kaggle e UCI Machine Learning Repository forniscono un comodo punto di partenza. I dataset preesistenti sono preziosi per ricercatori, sviluppatori e aziende. Uno dei loro principali vantaggi è l'efficienza dei costi. Creare un set di dati da zero può richiedere molto tempo ed essere costoso, quindi accedere a dati già pronti può far risparmiare risorse significative. Inoltre, molti set di dati, come ImageNet, sono diventati parametri di riferimento standard nella comunità di apprendimento automatico, consentendo confronti di prestazioni coerenti tra diversi modelli e algoritmi. Questa disponibilità di dati significa che gli esperimenti possono essere avviati immediatamente senza ritardi nella raccolta e nella preelaborazione dei dati. In un campo in rapida evoluzione come il ML, questa praticità è importante.

La garanzia di qualità che deriva dai dataset preesistenti più diffusi è importante da considerare perché diversi set di dati contengono errori. Ad esempio, nel set di dati ImageNet è stato riscontrato oltre il 6,4% di errori. Dato il loro uso diffuso, la comunità spesso identifica e corregge eventuali errori o distorsioni in questi set di dati. Questa garanzia è particolarmente utile per studenti e nuovi arrivati nel campo, in quanto possono concentrarsi sull'apprendimento e sulla sperimentazione senza preoccuparsi dell'integrità dei dati. La documentazione di supporto che spesso accompagna i set di dati esistenti è inestimabile, sebbene ciò si applichi generalmente solo a quelli ampiamente utilizzati. Una buona documentazione fornisce approfondimenti sul processo di raccolta dati e sulle definizioni delle variabili e talvolta offre persino prestazioni del modello di base. Queste informazioni non solo aiutano la comprensione, ma promuovono anche la riproducibilità nella ricerca, un pilastro dell'integrità scientifica; attualmente, c'è una crisi attorno al miglioramento della riproducibilità nei sistemi di apprendimento automatico. Quando altri ricercatori hanno accesso agli stessi dati, possono convalidare i risultati, testare nuove ipotesi o applicare metodologie diverse, consentendoci così di basarci più rapidamente sul lavoro reciproco.

Sebbene piattaforme come Kaggle e UCI Machine Learning Repository siano risorse inestimabili, è essenziale comprendere il contesto in cui sono stati raccolti i dati. I ricercatori dovrebbero fare attenzione al potenziale "overfitting" quando utilizzano set di dati popolari, poiché potrebbero essere stati addestrati più modelli su di essi, portando a metriche di prestazioni gonfiate. A volte, questi set di dati non riflettono i dati del mondo reale.

Negli ultimi anni, si è sviluppata una crescente consapevolezza dei problemi di "bias" [distorsione], validità e riproducibilità che possono esistere nei set di dati di apprendimento automatico. Figura 5.3 illustra un'altra preoccupazione critica: il potenziale di disallineamento quando si utilizza lo stesso set di dati per addestrare modelli diversi.

Figura 5.3: Addestrare modelli diversi con lo stesso set di dati. Fonte: (icons from left to right: Becris; Freepik; Freepik; Paul J; SBTS2018).

Come mostrato in Figura 5.3, addestrare più modelli utilizzando lo stesso set di dati può comportare un "disallineamento" tra i modelli e il mondo. Questo disallineamento crea un intero ecosistema di modelli che riflette solo un sottoinsieme ristretto di dati del mondo reale. Un tale scenario può portare a una generalizzazione limitata e a risultati potenzialmente distorti in varie applicazioni che utilizzano questi modelli.

5.3.2 Web Scraping

Il "web scraping" si riferisce a tecniche automatizzate per l'estrazione di dati dai siti Web. In genere comporta l'invio di richieste HTTP ai server Web, il recupero di contenuti HTML e l'analisi di tali contenuti per estrarre informazioni rilevanti. Gli strumenti e i framework più diffusi per il web scraping includono Beautiful Soup, Scrapy e Selenium. Questi strumenti offrono diverse funzionalità, dall'analisi dei contenuti HTML all'automazione delle interazioni con i browser Web, in particolare per i siti Web che caricano i contenuti in modo dinamico tramite JavaScript.

Il web scraping può raccogliere efficacemente grandi set di dati per l'addestramento di modelli di apprendimento automatico, in particolare quando i dati etichettati da esseri umani sono scarsi. Per la ricerca sulla visione artificiale, il web scraping consente la raccolta di enormi volumi di immagini e video. I ricercatori hanno utilizzato questa tecnica per creare set di dati influenti come ImageNet e OpenImages. Ad esempio, si potrebbero effettuare scraping di siti di e-commerce per accumulare foto di prodotti per il riconoscimento di oggetti o piattaforme di social media per raccogliere caricamenti di utenti per l'analisi facciale. Anche prima di ImageNet, il progetto LabelMe di Stanford ha raschiato (scraped) Flickr per oltre 63.000 immagini annotate che coprono centinaia di categorie di oggetti.

Oltre alla visione artificiale, lo scraping web supporta la raccolta di dati testuali per il linguaggio naturale. I ricercatori possono "raschiare" siti di notizie per dati di analisi del "sentiment", forum e siti di recensioni per la ricerca sui sistemi di dialogo o social media per la modellazione di argomenti. Ad esempio, i dati di training per il chatbot ChatGPT sono stati ottenuti tramite scraping di gran parte dell'Internet pubblico. I repository GitHub sono stati sottoposti a scraping per addestrare l'assistente di codifica Copilot AI di GitHub.

Il web scraping può anche raccogliere dati strutturati, come prezzi delle azioni, dati meteorologici o informazioni sui prodotti, per applicazioni analitiche. Una volta che i dati sono stati "raschiati", è essenziale archiviarli in modo strutturato, spesso utilizzando database o data warehouse. Una corretta gestione dei dati garantisce l'usabilità dei dati raccolti per analisi e applicazioni future.

Tuttavia, mentre il web scraping offre numerosi vantaggi, ci sono limitazioni significative e considerazioni etiche da sostenere. Non tutti i siti Web consentono lo scraping e la violazione di queste restrizioni può portare a ripercussioni legali. Anche lo scraping di materiale protetto da copyright o comunicazioni private è immorale e potenzialmente illegale. Il web scraping etico impone l'aderenza al file 'robots.txt' di un sito web, che delinea le sezioni del sito a cui è possibile accedere e che possono essere scansionate dai bot automatizzati.

Per scoraggiare lo scraping automatizzato, molti siti web implementano limiti di velocità. Se un bot invia troppe richieste in un breve periodo, potrebbe essere temporaneamente bloccato, limitando la velocità di accesso ai dati. Inoltre, la natura dinamica dei contenuti web implica che i dati estratti a intervalli diversi potrebbero richiedere maggiore coerenza, ponendo sfide per gli studi a lungo termine. Tuttavia, ci sono tendenze emergenti come la Web Navigation in cui gli algoritmi di apprendimento automatico possono navigare automaticamente nel sito web per accedere ai contenuti dinamici.

Il volume di dati pertinenti disponibili per lo scraping potrebbe essere limitato per argomenti di nicchia. Ad esempio, mentre lo scraping per argomenti comuni come immagini di gatti e cani potrebbe produrre dati abbondanti, la ricerca di condizioni mediche rare potrebbe essere meno fruttuosa. Inoltre, i dati ottenuti tramite scraping sono spesso non strutturati e rumorosi, il che richiede un'accurata pre-elaborazione e pulizia. È fondamentale comprendere che non tutti i dati raccolti saranno di alta qualità o accuratezza. L'impiego di metodi di verifica, come il riferimento incrociato con fonti alternative di dati, può migliorare l'affidabilità dei dati.

Quando si esegue lo scraping di dati personali, sorgono problemi di privacy, sottolineando la necessità di anonimizzazione. Pertanto, è fondamentale aderire ai "Termini del Servizio" di un sito Web, limitare la raccolta di dati a quelli di dominio pubblico e garantire l'anonimato di tutti i dati personali acquisiti.

Mentre il web scraping può essere un metodo scalabile per accumulare grandi set di dati di training per sistemi di intelligenza artificiale, la sua applicabilità è limitata a tipi di dati specifici. Ad esempio, il web scraping rende più complessa la ricerca di dati per unità di misura inerziali (IMU) per il riconoscimento dei gesti. Al massimo, si può effettuare lo scraping di un set di dati esistente.

La raccolta dal Web può produrre dati incoerenti o imprecisi. Ad esempio, la foto in Figura 5.4 viene visualizzata quando si cerca "semaforo" su Google Images. È un'immagine del 1914 che mostra semafori obsoleti, che sono anche appena distinguibili a causa della scarsa qualità dell'immagine. Questo può essere problematico per i set di dati estratti dal Web, poiché lo inquina con campioni di dati non applicabili (vecchi).

Figura 5.4: Un'immagine di vecchi semafori (1914). Fonte: Vox.

Esercizio 5.2: Web Scraping

Scoprire la potenza del web scraping con Python usando librerie come Beautiful Soup e Pandas. Questo esercizio estrarrà la documentazione Python per i nomi e le descrizioni delle funzioni ed esplorerà le statistiche dei giocatori NBA. Alla fine, si avranno le competenze per estrarre e analizzare dati da siti Web reali. Pronti all'immersione? Accedere al notebook Google Colab qui sotto e iniziare a fare pratica!

Open in Colab

5.3.3 Crowdsourcing

Il crowdsourcing per i dataset è la pratica di ottenere dati utilizzando i servizi di molte persone, sia da una comunità specifica che dal pubblico in generale, in genere tramite Internet. Invece di affidarsi a un piccolo team o a un'organizzazione specifica per raccogliere o etichettare i dati, il crowdsourcing sfrutta lo sforzo collettivo di un vasto gruppo distribuito di partecipanti. Servizi come Amazon Mechanical Turk consentono la distribuzione di attività di annotazione a una forza lavoro ampia e diversificata. Questo facilita la raccolta di etichette per attività complesse come l'analisi del "sentiment" o il riconoscimento delle immagini che richiedono il giudizio umano.

Il crowdsourcing è emerso come un approccio efficace per la raccolta di dati e la risoluzione dei problemi. Uno dei principali vantaggi del crowdsourcing è la scalabilità: distribuendo le attività a un ampio pool globale di collaboratori su piattaforme digitali, i progetti possono elaborare rapidamente enormi volumi di dati. Ciò rende il crowdsourcing ideale per l'etichettatura, la raccolta e l'analisi di dati su larga scala.

Inoltre, il crowdsourcing attinge a un gruppo eterogeneo di partecipanti, apportando un'ampia gamma di prospettive, intuizioni culturali e capacità linguistiche che possono arricchire i dati e migliorare la risoluzione creativa dei problemi in modi che un gruppo più omogeneo potrebbe non fare. Poiché il crowdsourcing attinge da un vasto pubblico oltre i canali tradizionali, è più conveniente rispetto ai metodi convenzionali, soprattutto per microattività più semplici.

Le piattaforme di crowdsourcing consentono anche una grande flessibilità, poiché i parametri delle attività possono essere modificati in tempo reale in base ai risultati iniziali. Ciò crea un ciclo di feedback per miglioramenti iterativi al processo di raccolta dati. I lavori complessi possono essere suddivisi in microattività e distribuiti a più persone, con risultati convalidati in modo incrociato assegnando versioni ridondanti della stessa attività. Se gestito in modo ponderato, il crowdsourcing consente il coinvolgimento della comunità attorno a un progetto collaborativo, in cui i partecipanti trovano una ricompensa nel contribuire.
Tuttavia, mentre il crowdsourcing offre numerosi vantaggi, è essenziale affrontarlo con una strategia chiara. Mentre fornisce l'accesso a un set diversificato di annotatori, introduce anche variabilità nella qualità delle annotazioni. Inoltre, piattaforme come Mechanical Turk potrebbero non sempre catturano uno spettro demografico completo; spesso, gli individui esperti di tecnologia sono sovra-rappresentati, mentre i bambini e gli anziani potrebbero essere sotto-rappresentati. Fornire istruzioni chiare e formazione per gli annotatori è fondamentale. Controlli periodici e convalide dei dati etichettati aiutano a mantenere la qualità. Ciò si ricollega all'argomento della chiara definizione del problema di cui abbiamo discusso in precedenza. Il crowdsourcing per i set di dati richiede anche una particolare attenzione alle considerazioni etiche. È fondamentale assicurarsi che i partecipanti siano informati su come verranno utilizzati i loro dati e che la loro privacy sia protetta. Il controllo di qualità tramite protocolli dettagliati, trasparenza nell'approvvigionamento e verifica è essenziale per garantire risultati affidabili.
Per TinyML, il crowdsourcing può presentare alcune sfide uniche. I dispositivi TinyML sono altamente specializzati per attività particolari entro vincoli rigorosi. Di conseguenza, i dati di cui hanno bisogno tendono a essere molto specifici. Ottenere tali dati specializzati da un pubblico generico può essere difficile tramite crowdsourcing. Ad esempio, le applicazioni TinyML spesso si basano su dati raccolti da determinati sensori o hardware. Il crowdsourcing richiederebbe ai partecipanti di avere accesso a dispositivi molto specifici e coerenti, come i microfoni, con le stesse frequenze di campionamento. Queste sfumature hardware presentano ostacoli anche per semplici attività audio come l'individuazione di parole chiave. Oltre all'hardware, i dati stessi necessitano di elevata granularità e qualità, dati i limiti di TinyML. Può essere difficile garantire ciò quando si fa crowdsourcing da chi non ha familiarità con il contesto e i requisiti dell'applicazione. Ci sono anche potenziali problemi relativi alla privacy, alla raccolta in tempo reale, alla standardizzazione e alle competenze tecniche. Inoltre, la natura ristretta di molte attività TinyML semplifica l'etichettatura accurata dei dati con la giusta comprensione. I partecipanti potrebbero aver bisogno di un contesto completo per fornire annotazioni affidabili.
Pertanto, mentre il crowdsourcing può funzionare bene in molti casi, le esigenze specializzate di TinyML introducono sfide uniche per i dati. È richiesta un'attenta pianificazione per linee guida, targeting e controllo di qualità. Per alcune applicazioni, il crowdsourcing potrebbe essere fattibile, ma altre potrebbero richiedere più lavoro per la raccolta dati più mirati per ottenere dati di training pertinenti e di alta qualità.

5.3.4 Dati Sintetici

La generazione di dati sintetici può essere una soluzione preziosa per affrontare le limitazioni della raccolta dati. Figura 5.5 illustra come funziona questo processo: i dati sintetici vengono uniti ai dati storici per creare un set di dati più ampio e diversificato per l'addestramento del modello.

Figura 5.5: Aumento delle dimensioni dei dati di training con la generazione di dati sintetici. Fonte: AnyLogic.

Come mostrato nella figura, i dati sintetici implicano la creazione di informazioni che non sono state originariamente catturate o osservate, ma vengono generate utilizzando algoritmi, simulazioni o altre tecniche per assomigliare ai dati del mondo reale. Questo approccio è diventato particolarmente prezioso in campi in cui i dati del mondo reale sono scarsi, costosi o eticamente difficili da ottenere, come nelle applicazioni TinyML. Varie tecniche, tra cui le "Generative Adversarial Networks (GAN)", possono produrre dati sintetici di alta qualità quasi indistinguibili dai dati reali. Questi metodi hanno fatto notevoli progressi, rendendo la generazione di dati sintetici sempre più realistica e affidabile.
Potrebbe essere necessario disporre di più dati del mondo reale per l'analisi o l'addestramento di modelli di apprendimento automatico in molti domini, in particolare quelli emergenti. I dati sintetici possono colmare questa lacuna producendo grandi volumi di dati che imitano scenari del mondo reale. Ad esempio, rilevare il suono di un vetro che si rompe potrebbe essere difficile nelle applicazioni di sicurezza in cui un dispositivo TinyML sta cercando di identificare le effrazioni. La raccolta di dati del mondo reale richiederebbe la rottura di numerose finestre, il che è poco pratico e costoso.
Inoltre, avere un set di dati diversificato è fondamentale nell'apprendimento automatico, in particolare nel deep learning. I dati sintetici possono aumentare i set di dati esistenti introducendo varianti, migliorando così la robustezza dei modelli. Ad esempio, SpecAugment è un'eccellente tecnica di aumento dei dati per i sistemi di "Automatic Speech Recognition" (ASR).
Anche la privacy e la riservatezza sono grandi problemi. I set di dati contenenti informazioni sensibili o personali sollevano problemi di privacy quando vengono condivisi o utilizzati. I dati sintetici, essendo generati artificialmente, non hanno questi legami diretti con individui reali, consentendo un utilizzo più sicuro preservando al contempo le proprietà statistiche essenziali.
La generazione di dati sintetici, in particolare una volta stabiliti i meccanismi di generazione, può essere un'alternativa più conveniente. I dati sintetici eliminano la necessità di rompere più finestre per raccogliere dati rilevanti nello scenario applicativo di sicurezza di cui sopra. Molti casi d'uso embedded riguardano situazioni uniche, come gli impianti di produzione, che sono difficili da simulare. I dati sintetici consentono ai ricercatori il controllo completo sul

processo di generazione dei dati, consentendo la creazione di scenari o condizioni specifici che sono difficili da catturare nella vita reale.
Sebbene i dati sintetici offrano numerosi vantaggi, è essenziale utilizzarli giudiziosamente. Bisogna fare attenzione a garantire che i dati generati rappresentino accuratamente le distribuzioni sottostanti del mondo reale e non introducano distorsioni indesiderate.

Esercizio 5.3: Dati Sintetici

Scopriamo la generazione di dati sintetici utilizzando le Generative Adversarial Network (GAN) su dati tabellari. Adotteremo un approccio pratico, immergendoci nel funzionamento del modello CTGAN e applicandolo al set di dati Synthea dal dominio sanitario. Dalla pre-elaborazione dei dati al training e valutazione del modello, procederemo passo dopo passo, imparando come creare dati sintetici, valutarne la qualità e sbloccare il potenziale delle GAN per l'aumento dei dati e le applicazioni del mondo reale.

Open in Colab

5.4 Archiviazione Dati

L'approvvigionamento e l'archiviazione dei dati vanno di pari passo e i dati devono essere archiviati in un formato che faciliti l'accesso e l'elaborazione. A seconda del caso d'uso, possono essere utilizzati vari tipi di sistemi di archiviazione dati per archiviare i set di dati. Alcuni esempi sono mostrati in Tabella 5.1.

Tabella 5.1: Panoramica comparativa del database, del data warehouse e del data lake.

Database	Data Warehouse	Data Lake
Scopo	Operativo e transazionale	Analitico
Tipo di dati	Strutturato	Strutturato, semi-strutturato e/o non strutturato
Scala	Da piccoli a grandi volumi di dati	Grandi volumi di dati integrati Grandi volumi di dati diversi
Esempi	MySQL	Google BigQuery, Amazon Redshift, Microsoft Azure Synapse, Google Cloud Storage, AWS S3, Azure Data Lake Storage

I dati archiviati sono spesso accompagnati da metadati, definiti come "dati sui dati". Forniscono informazioni contestuali dettagliate sui dati, come mezzi di creazione dei dati, ora di creazione, licenza di utilizzo dei dati allegata, ecc. Figura 5.6 illustra i pilastri chiave della raccolta dati e i metodi associati, evidenziando l'importanza della gestione dei dati strutturati. Ad esempio, Hugging Face ha implementato le Dataset Cards per promuovere un utilizzo responsabile dei dati. Queste "card" [schede], che si allineano al pilastro della documentazione mostrato in Figura 5.6, consentono ai creatori di dataset di rivelare potenziali bias e di istruire gli utenti sui contenuti e le limitazioni di un dataset.
Le "dataset card" forniscono un contesto importante sull'utilizzo appropriato del dataset evidenziando "bia" [pregiudizi] e altri dettagli importanti. Avere questo tipo di metadati strutturati può anche consentire un rapido recupero, allineandosi ai principi di gestione efficiente dei dati illustrati nella figura. Una volta sviluppato e distribuito il modello sui dispositivi edge, i sistemi di storage possono continuare a memorizzare dati in arrivo, aggiornamenti del modello o risultati analitici, utilizzando potenzialmente metodi da più pilastri mostrati in Figura 5.6. Questo processo continuo di raccolta e gestione dei dati garantisce che il modello rimanga aggiornato e pertinente nel suo ambiente operativo.

Figura 5.6: Pilastri della raccolta dati. Fonte: Alexsoft

Data Governance: Con una grande quantità di archiviazione dati, è anche fondamentale disporre di policy e pratiche (ad esempio, "governance" [gestione] dei dati) che aiutino a gestire i dati durante il loro ciclo di vita, dall'acquisizione allo smaltimento. La governance dei dati descrive il modo in cui i dati vengono gestiti e include l'adozione di decisioni chiave in merito al loro accesso e controllo. Figura 5.7 illustra i diversi domini coinvolti nella governance dei dati. Implica l'esercizio dell'autorità e l'assunzione di decisioni sui dati per mantenerne la qualità, garantire la conformità, mantenere la sicurezza e ricavarne valore. La governance dei dati è resa operativa sviluppando politiche, incentivi e sanzioni, coltivando una cultura che percepisce i dati come un bene prezioso. Procedure specifiche e autorità assegnate vengono implementate per salvaguardare la qualità dei dati e monitorarne l'utilizzo e i rischi correlati.

Figura 5.7: Una panoramica del framework di governance dei dati. Fonte: StarCIO..

La governance dei dati utilizza tre approcci integrativi: pianificazione e controllo, organizzativo e basato sul rischio.

- **L'approccio di pianificazione e controllo**, comune nell'IT, allinea business e tecnologia attraverso cicli annuali e continui aggiustamenti, concentrandosi su una governance verificabile e basata su policy.
- **L'approccio organizzativo** enfatizza la struttura, stabilendo ruoli autorevoli come Chief Data Officer e garantendo responsabilità e rendicontazione nella governance.
- **L'approccio basato sul rischio**, intensificato dai progressi dell'IA, si concentra sull'identificazione e la gestione dei rischi intrinseci nei dati e negli algoritmi. Affronta in particolare i problemi specifici dell'IA attraverso valutazioni regolari e strategie di gestione proattiva del rischio, consentendo azioni incidentali e preventive per mitigare gli impatti indesiderati degli algoritmi.

Ecco alcuni esempi di governance dei dati in diversi settori:

- **Medicina:** Gli Health Information Exchanges (HIE) [scambi di informazioni sanitarie] consentono la condivisione di informazioni sanitarie tra diversi operatori sanitari per migliorare l'assistenza ai pazienti. Implementano rigorose pratiche di governance dei dati per mantenere l'accuratezza, l'integrità, la privacy e la sicurezza dei dati, rispettando normative come l'Health Insurance Portability and Accountability Act (HIPAA). Le policy di governance assicurano che i dati dei pazienti siano condivisi solo con entità autorizzate e che i pazienti possano controllare l'accesso alle proprie informazioni.
- **Finanza:** Basilea III Framework è un quadro normativo internazionale per le banche. Garantisce che le banche stabiliscano policy, pratiche e responsabilità chiare per la gestione dei dati, assicurandone accuratezza, completezza e tempestività. Non solo consente alle banche di soddisfare la conformità normativa, ma previene anche le crisi finanziarie gestendo i rischi in modo più efficace.
- **Governo:** Le agenzie governative che gestiscono i dati dei cittadini, i registri pubblici e le informazioni amministrative implementano la governance dei dati per gestire i dati in modo trasparente e sicuro. Il sistema di previdenza sociale negli Stati Uniti e il sistema Aadhar in India sono buoni esempi di tali sistemi di governance.

Considerazioni speciali sull'archiviazione dei dati per TinyML

Formati di Archiviazione Audio Efficienti: I sistemi di individuazione delle parole chiave necessitano di formati di archiviazione audio specializzati per consentire una rapida ricerca delle parole chiave nei dati audio. I formati tradizionali come WAV e MP3 archiviano forme d'onda audio complete, che richiedono un'elaborazione estesa per la ricerca. L'individuazione delle parole chiave utilizza un archivio compresso ottimizzato per la ricerca basata su frammenti. Un approccio consiste nell'archiviazione di caratteristiche acustiche compatte anziché audio grezzo. Tale flusso di lavoro implicherebbe:

- **Estrazione di Caratteristiche Acustiche:** I coefficienti Mel-frequency cepstral (MFCC) rappresentano comunemente importanti caratteristiche audio.
- **Creazione di Embedding:** Gli "embedding" trasformano le caratteristiche acustiche estratte in spazi vettoriali continui, consentendo un'archiviazione dei dati più compatta e rappresentativa. Questa rappresentazione è essenziale per convertire dati ad alta dimensionalità, come l'audio, in un formato più gestibile ed efficiente per l'elaborazione e l'archiviazione.
- **Quantizzazione vettoriale:** Questa tecnica rappresenta dati ad alta dimensionalità, come gli embedding, con vettori a bassa dimensionalità, riducendo le esigenze di archiviazione. Inizialmente, un codebook viene generato dai dati di training per definire

un set di vettori di codice che rappresentano i vettori di dati originali. Successivamente, ogni vettore di dati viene abbinato alla "codeword" più vicina in base al codebook, garantendo una perdita minima di informazioni.

- **Archiviazione sequenziale:** L'audio viene frammentato in frame brevi e le feature caratteristiche quantizzate (o embedded) per ogni frame vengono archiviate in sequenza per mantenere l'ordine temporale, preservando la coerenza e il contesto dei dati audio.

Questo formato consente di decodificare le feature frame per frame per la corrispondenza delle parole chiave. La ricerca delle caratteristiche è più rapida della decompressione dell'audio completo.

Selective Network Output Storage: [Archiviazione selettiva dell'output di rete] Un'altra tecnica per ridurre l'archiviazione consiste nell'eliminare le caratteristiche audio intermedie archiviate durante l'addestramento ma non richieste durante l'inferenza. La rete viene eseguita su audio completo durante l'addestramento. Tuttavia, solo gli output finali vengono archiviati durante l'inferenza.

5.5 Elaborazione dei Dati

Il "Data processing" elaborazione dei dati si riferisce ai passaggi necessari per trasformare i dati grezzi in un formato adatto per l'inserimento negli algoritmi di apprendimento automatico. È una fase cruciale in qualsiasi flusso di lavoro ML, ma spesso trascurata. Con un'elaborazione dei dati adeguata, è probabile che i modelli ML raggiungano prestazioni ottimali. Figura 5.8 mostra una ripartizione dell'allocazione del tempo di uno scienziato dei dati, evidenziando la parte significativa spesa per la pulizia e l'organizzazione dei dati (%60).

Figura 5.8: Ripartizione delle attività dei "Data scientist" in base al tempo impiegato. Fonte: Forbes.

Una corretta pulizia dei dati è un passaggio cruciale che influisce direttamente sulle prestazioni del modello. I dati del mondo reale sono spesso sporchi, contengono errori, valori mancanti, rumore, anomalie e incongruenze. La pulizia dei dati comporta il rilevamento e la correzione di questi problemi per preparare dati di alta qualità per la modellazione. Selezionando attentamente le tecniche appropriate, i data scientist possono migliorare l'accuratezza del modello, ridurre l'overfitting e addestrare gli algoritmi per apprendere pattern più solidi. Nel complesso, un'elaborazione dei dati ponderata consente ai sistemi di apprendimento automatico di scoprire meglio le informazioni e di fare previsioni dai dati del mondo reale.

I dati spesso provengono da fonti diverse e possono essere non strutturati o semi-strutturati. Pertanto, elaborarli e standardizzarli è essenziale, assicurando che aderiscano a un formato uniforme. Tali trasformazioni possono includere:

- Normalizzazione di variabili numeriche
- Codifica di variabili categoriali
- Utilizzo di tecniche come la riduzione della dimensionalità

La convalida dei dati svolge un ruolo più ampio rispetto alla garanzia di aderenza a determinati standard, come impedire che i valori di temperatura scendano sotto lo zero assoluto. Questi problemi si verificano in TinyML perché i sensori potrebbero funzionare male o produrre temporaneamente letture errate; tali transienti non sono rari. Pertanto, è fondamentale rilevare gli errori nei dati in anticipo prima che si propaghino attraverso la pipeline dei dati. Rigorosi processi di convalida, tra cui la verifica delle pratiche di annotazione iniziali, il rilevamento di valori anomali e la gestione dei valori mancanti tramite tecniche come l'imputazione della media, contribuiscono direttamente alla qualità dei set di dati. Ciò, a sua volta, influisce sulle prestazioni, la correttezza e la sicurezza dei modelli addestrati su di essi.

Diamo un'occhiata a Figura 5.9 per un esempio di pipeline di elaborazione dei dati. Nel contesto di TinyML, il Multilingual Spoken Words Corpus (MSWC) è un esempio di pipeline di elaborazione dei dati, flussi di lavoro sistematici e automatizzati per la trasformazione, l'archiviazione e l'elaborazione dei dati. I dati di input (che sono una raccolta di brevi registrazioni) attraversano diverse fasi di elaborazione, come l'allineamento audio-parola e l'estrazione di parole chiave.

MSWC semplifica il flusso di dati, dai dati grezzi ai set di dati utilizzabili, le pipeline di dati migliorano la produttività e facilitano lo sviluppo rapido di modelli di apprendimento automatico. MSWC è una raccolta ampia e in continua espansione di registrazioni audio di parole pronunciate in 50 lingue diverse, utilizzate collettivamente da oltre 5 miliardi di persone. Questo set di dati è destinato allo studio accademico e all'uso aziendale in aree come l'identificazione di parole chiave e la ricerca basata sul parlato. È concesso in licenza aperta con Creative Commons Attribution 4.0 per un ampio utilizzo.

Figura 5.9: Una panoramica della pipeline di elaborazione dati del Multilingual Spoken Words Corpus (MSWC). Fonte: Mazumder et al. (2021).

Il MSWC ha utilizzato un metodo di allineamento forzato per estrarre automaticamente singole registrazioni di parole per addestrare modelli di individuazione delle parole chiave dal progetto Common Voice, che presenta registrazioni a livello di frase in crowdsourcing. L'allineamento forzato si riferisce a metodi di lunga data nell'elaborazione del parlato che prevedono quando fenomeni del parlato come sillabe, parole o frasi iniziano e finiscono all'interno di una registrazione audio. Nei dati MSWC, le registrazioni in crowdsourcing

spesso presentano rumori di sottofondo, come elettricità statica e vento. A seconda dei requisiti del modello, questi rumori possono essere rimossi o mantenuti intenzionalmente. Mantenere l'integrità dell'infrastruttura dati è uno lavoro continuo. Ciò comprende archiviazione dei dati, sicurezza, gestione degli errori e rigoroso controllo delle versioni. Gli aggiornamenti periodici sono fondamentali, soprattutto in ambiti dinamici come l'individuazione delle parole chiave, per adattarsi alle tendenze linguistiche in evoluzione e alle integrazioni dei dispositivi.
C'è un boom nelle pipeline di elaborazione dati, comunemente presenti nelle toolchain delle operazioni ML, di cui parleremo nel capitolo MLOps. In breve, questi includono framework come MLOps di Google Cloud. Fornisce metodi per l'automazione e il monitoraggio in tutte le fasi della costruzione del sistema ML, tra cui integrazione, test, rilascio, distribuzione e gestione dell'infrastruttura. Diversi meccanismi si concentrano sull'elaborazione dati, parte integrante di questi sistemi.

Esercizio 5.4: Elaborazione dei Dati

Esploriamo due progetti significativi nell'elaborazione dei dati vocali e nell'apprendimento automatico. MSWC è un vasto set di dati audio con oltre 340.000 parole chiave e 23,4 milioni di esempi parlati di 1 secondo. Viene utilizzato in varie applicazioni come dispositivi voice-enabled e automazione dei call center. Il progetto Few-Shot Keyword Spotting introduce un nuovo approccio per l'individuazione delle parole chiave in diverse lingue, ottenendo risultati impressionanti con dati di addestramento minimi. Esamineremo il set di dati MSWC, impareremo come strutturarlo in modo efficace e poi addestreremo un modello di individuazione di parole chiave con la tecnica "few-shot" [https://www.ibm.com/it-it/topics/few-shot-learning]. Cominciamo!

Open in Colab

5.6 Etichettatura dei Dati

Il "Data labeling" etichettatura dei dati è importante per creare set di dati di training di alta qualità per modelli di apprendimento automatico. Le etichette forniscono informazioni di base, consentendo ai modelli di apprendere relazioni tra input e output desiderati. Questa sezione copre considerazioni chiave per la selezione di tipi di etichette, formati e contenuti per acquisire le informazioni necessarie per le attività. Discute approcci di annotazione comuni, dall'etichettatura manuale al crowdsourcing ai metodi assistiti dall'intelligenza artificiale, e le "best practice" per garantire la qualità delle etichette tramite formazione, linee guida e controlli di qualità. Sottolineiamo anche il trattamento etico degli annotatori umani. Viene anche esplorata l'integrazione dell'intelligenza artificiale per accelerare e aumentare l'annotazione umana. Comprendere le esigenze, le sfide e le strategie di etichettatura è essenziale per costruire dataset affidabili e utili per addestrare sistemi di apprendimento automatico performanti e affidabili.

5.6.1 Tipi di Etichette

Le etichette contengono informazioni su attività o concetti chiave. Figura 5.10 include alcuni tipi di etichette comuni: una "classification label" [etichetta di classificazione] viene utilizzata per categorizzare le immagini con etichette (etichettando un'immagine con "dog" [cane] e presenta un cane); un "bounding box" [riquadro delimitatore] identifica la posizione dell'oggetto (disegnando un riquadro attorno al cane); una "segmentation map" [mappa di segmentazione] classifica gli oggetti a livello di pixel (evidenziando il cane con un colore distinto); una "caption" [didascalia] fornisce annotazioni descrittive (descrivendo le azioni, la posizione, il colore, ecc. del cane); e una "transcript" [trascrizione] denota il contenuto

audio. La scelta del formato dell'etichetta dipende dal caso d'uso e dai vincoli di risorse, poiché etichette più dettagliate richiedono un lavoro maggiore per la raccolta (Johnson-Roberson et al. 2017).

Label Type	Input Type	Output Type
Classification Label		"Dog", "Blanket", "No cat"
Bounding Box		
Segmentation Map		
Caption		"A dog curls up on a spotted purple blanket."
Transcript		"Once upon a time, a dog was curled up on a spotted purple blanket …"

Figura 5.10: Una panoramica dei tipi di etichette comuni.

A meno che non si concentri sull'apprendimento auto-supervisionato, un set di dati fornirà probabilmente etichette che affrontano una o più attività di interesse. Date le loro limitazioni di risorse uniche, i creatori di set di dati devono considerare quali informazioni le etichette dovrebbero catturare e come possono ottenere praticamente le etichette necessarie. I creatori devono prima decidere quali tipi di etichette di contenuto dovrebbero catturare. Ad esempio, un creatore interessato al rilevamento delle auto vorrebbe etichettare le auto nel suo dataset. Tuttavia, potrebbe anche considerare se raccogliere simultaneamente etichette per altre attività per cui il set di dati potrebbe essere potenzialmente utilizzato, come il rilevamento dei pedoni.

Inoltre, gli annotatori possono fornire metadati per le informazioni su come il set di dati rappresenta diverse caratteristiche di interesse (cfr. Sezione 5.9). Il dataset Common Voice, ad esempio, include vari tipi di metadati che forniscono informazioni sugli oratori, sulle registrazioni e sulla qualità del set di dati per ciascuna lingua rappresentata (Ardila et al. 2020). Includono suddivisioni demografiche che mostrano il numero di registrazioni per fascia di età e genere del parlante. Questo ci consente di vedere chi ha contribuito alle registrazioni per ogni lingua. Includono anche statistiche come la durata media delle registrazioni e il numero totale di ore di registrazioni convalidate. Queste forniscono informazioni sulla natura e le dimensioni dei set di dati per ogni lingua.

Inoltre, le metriche di controllo qualità come la percentuale di registrazioni convalidate sono utili per sapere quanto siano completi e puliti i set di dati. I metadati includono anche suddivisioni demografiche normalizzate scalate al 100% per il confronto tra le lingue. Questo evidenzia le differenze di rappresentazione tra lingue con risorse più elevate e più basse.

Successivamente, i creatori devono determinare il formato di tali etichette. Ad esempio, un creatore interessato al rilevamento delle auto potrebbe scegliere tra etichette di classificazione binaria che indicano se è presente un'auto, riquadri di delimitazione che

mostrano le posizioni generali di tutte le auto o etichette di segmentazione pixel per pixel che mostrano la posizione esatta di ogni auto. La scelta del formato dell'etichetta può dipendere dal caso d'uso e dai vincoli di risorse, poiché le etichette più dettagliate sono in genere più costose e richiedono più tempo per essere acquisite.

5.6.2 Metodi di Annotazione

Gli approcci comuni all'annotazione includono etichettatura manuale, crowdsourcing e tecniche semi-automatiche. L'etichettatura manuale da parte di esperti produce alta qualità ma necessita di maggiore scalabilità. Il crowdsourcing consente ai non esperti di distribuire annotazioni, spesso tramite piattaforme dedicate (Sheng e Zhang 2019). Metodi debolmente supervisionati e programmatici possono ridurre il lavoro manuale generando etichette in modo euristico o automatico (Ratner et al. 2018).

Dopo aver deciso il contenuto e il formato desiderati per le etichette, i creatori iniziano il processo di annotazione. Per raccogliere un gran numero di etichette da annotatori umani, i creatori si affidano spesso a piattaforme di annotazione dedicate, che possono metterli in contatto con team di annotatori umani. Quando utilizzano queste piattaforme, i creatori potrebbero aver bisogno di maggiori informazioni sui background degli annotatori e sui livelli di esperienza con argomenti di interesse. Tuttavia, alcune piattaforme offrono l'accesso ad annotatori con competenze specifiche (ad esempio, medici).

Esercizio 5.5: Etichette Autoprodotte

Esploriamo Wake Vision, un set di dati completo progettato per il rilevamento di persone con TinyML. Questo set di dati deriva da un set di dati più ampio e generico, Open Images (Kuznetsova et al. 2020), e specificamente adattato per il rilevamento binario di persone. Il processo di trasformazione comporta il filtraggio e la rietichettatura delle etichette e dei riquadri di delimitazione esistenti in Open Images utilizzando una pipeline automatizzata. Questo metodo non solo consente di risparmiare tempo e risorse, ma garantisce anche che il set di dati soddisfi i requisiti specifici delle applicazioni TinyML.

Inoltre, generiamo metadati per confrontare la correttezza e la robustezza dei modelli in scenari difficili.

Cominciamo!

Open in Colab

5.6.3 Garantire la Qualità dell'Etichetta

Non vi è alcuna garanzia che le etichette dei dati siano effettivamente corrette. Figura 5.11 mostra alcuni esempi di casi di etichettatura rigida: alcuni errori derivano da immagini sfocate che le rendono difficili da identificare (l'immagine della rana), e altri derivano da una mancanza di conoscenza del dominio (il caso della cicogna nera). È possibile che nonostante le migliori istruzioni fornite agli etichettatori, etichettino ancora in modo errato alcune immagini (Northcutt, Athalye, e Mueller 2021). Strategie come controlli di qualità, formazione degli annotatori e raccolta di più etichette per ciascun elemento possono aiutare a garantire la qualità delle etichette. Per attività ambigue, più annotatori possono aiutare a identificare i punti dati controversi e quantificare i livelli di disaccordo.

Figura 5.11: Alcuni esempi di casi di etichettatura rigida. Fonte: Northcutt, Athalye, e Mueller (2021).

Quando si lavora con annotatori umani, è importante offrire un compenso equo e dare priorità al trattamento etico, poiché gli annotatori possono essere sfruttati o danneggiati durante il processo di etichettatura (Perrigo, 2023). Ad esempio, se è probabile che un set di dati contenga contenuti inquietanti, gli annotatori potrebbero trarre vantaggio dall'avere la possibilità di visualizzare le immagini in scala di grigi (Google, s.d.).

5.6.4 Annotazione assistita dall'intelligenza artificiale

Il ML ha una domanda insaziabile di dati. Pertanto, sono necessari più dati. Ciò solleva la questione di come possiamo ottenere più dati etichettati. Invece di generare e curare sempre i dati manualmente, possiamo fare affidamento sui modelli di intelligenza artificiale esistenti per etichettare i set di dati in modo più rapido ed economico, anche se spesso con una qualità inferiore rispetto all'annotazione umana. Questo può essere fatto in vari modi come mostrato in Figura 5.12, tra cui i seguenti:

- **Pre-annotazione:** I modelli di intelligenza artificiale possono generare etichette preliminari per un set di dati utilizzando metodi come l'apprendimento semi-supervisionato (Chapelle, Scholkopf, e Zien 2009), che gli esseri umani possono poi esaminare e correggere. Questo può far risparmiare una notevole quantità di tempo, soprattutto per set di dati di grandi dimensioni.
- **Apprendimento attivo:** I modelli di intelligenza artificiale possono identificare i dati più informativi in un dataset, che possono quindi essere riordinati per priorità per l'annotazione umana. Questo può aiutare a migliorare la qualità del set di dati etichettato riducendo al contempo il tempo di annotazione complessivo.
- **Controllo qualità:** I modelli di intelligenza artificiale possono identificare e segnalare potenziali errori nelle annotazioni umane, contribuendo a garantire l'accuratezza e la coerenza del set di dati etichettato.

Figura 5.12: Strategie per acquisire ulteriori dati di addestramento etichettati. Fonte: Standford AI Lab.

Ecco alcuni esempi di come l'annotazione assistita dall'intelligenza artificiale è stata proposta come utile:

- **Immagini mediche:** L'annotazione assistita dall'intelligenza artificiale etichetta le immagini mediche, come scansioni MRI (Magnetic Resonance Imaging) e raggi X (R. Krishnan, Rajpurkar, e Topol 2022). Annotare attentamente i set di dati medici è estremamente impegnativo, soprattutto su larga scala, poiché gli esperti del settore sono scarsi e diventano costosi. Ciò può aiutare ad addestrare i modelli di intelligenza artificiale per diagnosticare malattie e altre condizioni mediche in modo più accurato ed efficiente.
- **Auto a guida autonoma:** L'annotazione assistita dall'intelligenza artificiale viene utilizzata per etichettare immagini e video di auto a guida autonoma. Ciò può aiutare ad addestrare i modelli di intelligenza artificiale per identificare oggetti sulla strada, come altri veicoli, pedoni e segnali stradali.
- **Social media:** L'annotazione assistita dall'intelligenza artificiale etichetta i post sui social media come immagini e video. Ciò può aiutare ad addestrare i modelli di intelligenza artificiale a identificare e classificare diversi tipi di contenuti, come notizie, pubblicità e post personali.

5.7 Controllo della Versione dei Dati

I sistemi di produzione sono costantemente inondati da volumi di dati fluttuanti e in aumento, che determinano la rapida comparsa di numerose repliche di dati. Questi dati in aumento servono come base per l'addestramento di modelli di apprendimento automatico. Ad esempio, un'azienda di vendita globale impegnata nella previsione delle vendite riceve continuamente dati sul comportamento dei consumatori. Allo stesso modo, i sistemi sanitari che formulano modelli predittivi per la diagnosi delle malattie acquisiscono costantemente nuovi dati sui pazienti. Le applicazioni TinyML, come l'individuazione delle parole chiave, sono molto affamate di dati per quanto riguarda la quantità di dati generati. Di conseguenza, è fondamentale un monitoraggio meticoloso delle versioni dei dati e delle prestazioni del modello corrispondente.
Il "Data Version Control" [controllo delle versioni dei dati] offre una metodologia strutturata per gestire in modo efficiente alterazioni e versioni di set di dati. Facilita il monitoraggio delle modifiche, conserva più versioni e garantisce riproducibilità e tracciabilità nei progetti incentrati sui dati. Inoltre, il controllo delle versioni dei dati offre la versatilità di rivedere e utilizzare versioni specifiche in base alle necessità, garantendo che ogni fase dell'elaborazione dei dati e dello sviluppo del modello possa essere riesaminata e verificata in modo preciso e semplice. Ha una varietà di usi pratici -

Gestione del Rischio: Il controllo della versione dei dati consente trasparenza e responsabilità monitorando le versioni del set di dati.

Collaborazione ed Efficienza: Un facile accesso a diverse versioni del set di dati in un unico posto può migliorare la condivisione dei dati di controllo specifici e consentire una collaborazione efficiente.

Riproducibilità: Il controllo della versione dei dati consente di monitorare le prestazioni dei modelli riguardanti diverse versioni dei dati, e quindi di abilitare la riproducibilità.

Concetti Chiave

- **Commit:** È un'istantanea immutabile dei dati in un momento specifico, che rappresenta una versione univoca. Ogni commit è associato a un identificatore univoco per consentire

- **Branch:** I "branch" [rami] consentono a sviluppatori e specialisti dei data di discostarsi dalla linea di sviluppo principale e continuare a lavorare in modo indipendente senza influenzare altri rami. Ciò è particolarmente utile quando si sperimentano nuove funzionalità o modelli, consentendo sviluppo e sperimentazione paralleli senza il rischio di danneggiare il ramo principale stabile.
- **Merge:** I "Merge" [unioni] aiutano a integrare le modifiche da rami diversi mantenendo l'integrità dei dati.

Con il controllo della versione dei dati in atto, possiamo tracciare le modifiche mostrate in Figura 5.13, riprodurre i risultati precedenti ripristinando le versioni precedenti e collaborare in modo sicuro ramificando e isolando le modifiche.

Figura 5.13: Data versioning.

Sistemi di Data Version Control più Diffusi

[DVC]: È l'acronimo di Data Version Control ed è uno strumento open source e leggero che funziona su Git Hub e supporta tutti i tipi di formati di dati. Può integrarsi perfettamente nel flusso di lavoro se Git viene utilizzato per gestire il codice. Cattura le versioni dei dati e dei modelli nei commit Git mentre li archivia in locale o sul cloud (ad esempio, AWS, Google Cloud, Azure). Questi dati e modelli (ad esempio, artefatti di ML) sono definiti nei file di metadati, che vengono aggiornati a ogni commit. Può consentire il monitoraggio delle metriche dei modelli su diverse versioni dei dati.

lakeFS: È uno strumento open source che supporta il controllo della versione dei dati sui "data lake". Supporta molte operazioni simili a git, come i "branch" e il "merge" dei dati, nonché il ripristino delle versioni precedenti dei dati. Ha anche una funzionalità UI unica, che semplifica notevolmente l'esplorazione e la gestione dei dati.

Git LFS: È utile per il controllo della versione dei dataset di dimensioni ridotte. Utilizza le funzionalità di "branch" e "merge" native di Git, ma è limitato nel tracciamento delle metriche, nel ripristino delle versioni precedenti o nell'integrazione con i "data lake".

5.8 Ottimizzazione dei Dati per l'IA Embedded

I creatori che lavorano su sistemi embedded potrebbero avere priorità insolite quando puliscono i loro dataset. Da un lato, i modelli potrebbero essere sviluppati per casi d'uso insolitamente specifici, che richiedono un filtraggio intensivo dei dataset. Mentre altri modelli

di linguaggio naturale possono essere in grado di trasformare qualsiasi discorso in testo, un modello per un sistema embedded può essere incentrato su un singolo compito limitato, come il rilevamento di una parola chiave. Di conseguenza, i creatori possono filtrare in modo aggressivo grandi quantità di dati perché devono affrontare un determinato compito. Un sistema di intelligenza artificiale embedded può anche essere legato a specifici dispositivi hardware o ambienti. Ad esempio, un modello video potrebbe dover elaborare immagini da un singolo tipo di telecamera, che verrà montata solo sui campanelli nei quartieri residenziali. In questo scenario, i creatori possono scartare le immagini se provengono da un diverso tipo di telecamera, mostrano il tipo sbagliato di scenario o sono state scattate dall'altezza o dall'angolazione sbagliate.
D'altra parte, ci si aspetta spesso che i sistemi di IA embedded forniscano prestazioni particolarmente accurate in contesti imprevedibili del mondo reale. Ciò può portare i creatori a progettare set di dati per rappresentare variazioni nei potenziali input e promuovere la robustezza del modello. Di conseguenza, possono definire un ambito ristretto per il loro progetto ma poi puntare a una copertura approfondita entro quei limiti. Ad esempio, i creatori del modello del campanello menzionato sopra potrebbero provare a coprire le variazioni nei dati derivanti da:

- Quartieri geograficamente, socialmente e architettonicamente diversi
- Diversi tipi di illuminazione artificiale e naturale
- Diverse stagioni e condizioni meteorologiche
- Ostruzioni (ad esempio gocce di pioggia o scatole di consegna che oscurano la visuale della telecamera)

Come descritto sopra, i creatori possono prendere in considerazione il crowdsourcing o la generazione sintetica di dati per includere queste varianti.

5.9 Trasparenza dei Dati

Fornendo una documentazione chiara e dettagliata, i creatori possono aiutare gli sviluppatori a capire come utilizzare al meglio i loro set di dati. Diversi gruppi hanno suggerito formati di documentazione standardizzati per i set di dati, come Data Cards (Pushkarna, Zaldivar, e Kjartansson 2022), datasheet (Gebru et al. 2021), data statement (Bender e Friedman 2018), o Data Nutrition Labels (Holland et al. 2020). Quando rilasciano un dataset, i creatori possono descrivere quali tipi di dati hanno raccolto, come li hanno raccolti ed etichettati e quali tipi di casi d'uso potrebbero essere adatti o meno al set di dati. Quantitativamente, potrebbe essere opportuno mostrare quanto bene il set di dati rappresenti gruppi diversi (ad esempio, gruppi di genere diversi, telecamere diverse).
Figura 5.14 mostra un esempio di una scheda dati per un set di dati di computer vision (CV). Include alcune informazioni di base sul set di dati e istruzioni su come utilizzarlo, inclusi i "bias" noti.

Open Images Extended - More Inclusively Annotated People (MIAP)

Dataset Download • Related Publication

This dataset was created for fairness research and fairness evaluations in person detection. This dataset contains 100,000 images sampled from Open Images V6 with additional annotations added. Annotations include the image coordinates of bounding boxes for each visible person. Each box is annotated with attributes for perceived gender presentation and age range presentation. It can be used in conjunction with Open Images V6.

Authorship

PUBLISHER(S)
Google LLC

INDUSTRY TYPE
Corporate - Tech

DATASET AUTHORS
Candice Schumann, Google, 2021
Susanna Ricco, Google, 2021
Utsav Prabhu, Google, 2021
Vittorio Ferrari, Google, 2021
Caroline Pantofaru, Google, 2021

FUNDING
Google LLC

FUNDING TYPE
Private Funding

DATASET CONTACT
open-images-extended@google.com

Motivations

DATASET PURPOSE(S)
Research Purposes
Machine Learning
Training, testing, and validation

KEY APPLICATION(S)
Machine Learning | Object Recognition | Machine Learning Fairness

PROBLEM SPACE
This dataset was created for fairness research and fairness evaluation with respect to person detection.
See accompanying article

PRIMARY MOTIVATION(S)
- Provide more complete ground-truth for bounding boxes around people.
- Provide a standard fairness evaluation set for the broader fairness community.

INTENDED AND/OR SUITABLE USE CASE(S)
- **ML Model Evaluation for:** Person detection, Fairness evaluation
- **ML Model Training for:** Person detection, Object detection

Additionally:
- **Person detection:** Without specifying gender or age presentations
- **Fairness evaluations:** Over gender and age presentations
- **Fairness research:** Without building gender presentation or age classifiers

Use of Dataset

SAFETY OF USE
Conditional Use
There are some known unsafe applications.

UNSAFE APPLICATION(S)
⚠ Gender classification | Age classification

UNSAFE USE CASE(S)
This dataset **should not** be used to create gender or age classifiers. The intention of percieved gender and age labels is to capture gender and age presentation as assessed by a third party based on visual cues alone, rather than an individual's self-identified gender or actual age.

CONJUNCTIONAL USE
Safe to use with other datasets

KNOWN CONJUNCTIONAL DATASET(S)
- The data in this dataset can be combined with Open Images V6

KNOWN CONJUNCTIONAL USES
Analyzing bounding box annotations not annotated under the Open Images V6 procedure.

METHOD
Object Detection

SUMMARY
A person object detector can be trained using the Object Detection API in Tensorflow.

KNOWN CAVEATS
If this dataset is used in conjunction with the original Open Images dataset, negative examples of people should only be pulled from images with an explicit negative person image level label.

The dataset does not contain any examples not annotated as containing at least one person by the original Open Images annotation procedure.

METHOD
Fairness Evalutaion

SUMMARY
Fairness evaluations can be run over the splits of gender presentation and age presentation.

KNOWN CAVEATS
There still exists a gender presentation skew towards unknown and predominantly masculine, as well as an age presentation range skew towards middle.

Figura 5.14: Data card che descrive un dataset CV. Fonte: Pushkarna, Zaldivar, e Kjartansson (2022).

Tenere traccia della provenienza dei dati, essenzialmente le origini e il viaggio di ogni dato attraverso la pipeline dei dati, non è solo una buona pratica, ma un requisito essenziale per la qualità. La provenienza dei dati contribuisce in modo significativo alla trasparenza dei sistemi di machine learning. I sistemi trasparenti semplificano l'analisi dei dati, consentendo una migliore identificazione e rettifica di errori, bias o incongruenze. Ad esempio, se un modello di ML addestrato su dati medici non è performante in aree specifiche, tracciare la provenienza può aiutare a identificare se il problema riguarda i metodi di raccolta dati, i gruppi demografici rappresentati nei dati o altri fattori. Questo livello di trasparenza non aiuta solo a eseguire il debug del sistema, ma svolge anche un ruolo cruciale nel migliorare la qualità complessiva dei dati. Migliorando l'affidabilità e la credibilità del set di dati, la provenienza dei dati migliora anche le prestazioni del modello e la sua accettabilità tra gli utenti finali. Quando si produce la documentazione, i creatori devono anche specificare come gli utenti possono accedere al dataset e come questo verrà mantenuto nel tempo. Ad esempio, gli

utenti potrebbero dover sottoporsi a una formazione o ricevere un'autorizzazione speciale dai creatori prima di accedere a un set di dati di informazioni protette, come con molti dataset medici. In alcuni casi, gli utenti potrebbero non accedere direttamente ai dati. Devono invece inviare il loro modello per essere addestrato sull'hardware dei creatori del set di dati, seguendo una configurazione di apprendimento "federato" (Aledhari et al. 2020). I creatori possono anche descrivere per quanto tempo il dataset rimarrà accessibile, come gli utenti possono inviare feedback su eventuali errori che scoprono e se ci sono piani per aggiornare il set di dati.

Alcune leggi e normative promuovono anche la trasparenza dei dati attraverso nuovi requisiti per le organizzazioni:

- Il "General Data Protection Regulation (GDPR)" nell'Unione Europea: Stabilisce requisiti rigorosi per l'elaborazione e la protezione dei dati personali dei cittadini dell'UE. Impone policy sulla privacy in linguaggio semplice che spiegano chiaramente quali dati vengono raccolti, perché vengono utilizzati, per quanto tempo vengono archiviati e con chi vengono condivisi. Il GDPR impone inoltre che le informative sulla privacy debbano includere dettagli sulla base giuridica per l'elaborazione, i trasferimenti di dati, i periodi di conservazione, i diritti di accesso e cancellazione e le informazioni di contatto per i responsabili del trattamento dei dati.
- Il "California's Consumer Privacy Act" (CCPA): Il CCPA richiede policy sulla privacy chiare e diritti di esclusione per vendere dati personali. In modo significativo, stabilisce anche i diritti dei consumatori di essere interpellati per la divulgazione dei propri dati specifici. Le aziende devono fornire copie delle informazioni personali raccolte e dettagli su come vengono utilizzate, quali categorie vengono raccolte e cosa ricevono le terze parti. I consumatori possono identificare dati che ritengono debbano essere più accurati. La legge rappresenta un importante passo avanti nel potenziamento dell'accesso ai dati personali.

Garantire la trasparenza dei dati presenta diverse sfide, soprattutto perché richiede molto tempo e risorse finanziarie. I sistemi di dati sono anche piuttosto complessi e la trasparenza completa può richiedere tempo. La trasparenza completa può anche sopraffare i consumatori con troppi dettagli. Infine, è anche importante bilanciare il compromesso tra trasparenza e privacy.

5.10 Licenze

Molti dataset di alta qualità provengono da fonti proprietarie o contengono informazioni protette da copyright. Ciò introduce le licenze come una competenza legale impegnativa. Le aziende desiderose di addestrare sistemi di ML devono impegnarsi in trattative per ottenere licenze che garantiscano l'accesso legale a questi dataset. Inoltre, i termini delle licenze possono imporre restrizioni sulle applicazioni dei dati e sui metodi di condivisione. Il mancato rispetto di queste licenze può avere gravi conseguenze.

Ad esempio, ImageNet, uno dei dataset più ampiamente utilizzati per la ricerca sulla visione artificiale, è un caso emblematico. La maggior parte delle sue immagini è stata ottenuta da fonti online pubbliche senza esplicita autorizzazione, suscitando preoccupazioni etiche (Prabhu e Birhane, 2020). L'accesso al set di dati ImageNet per le aziende richiede la registrazione e l'adesione ai suoi termini di utilizzo, che limitano l'uso commerciale (ImageNet, 2021). I principali attori come Google e Microsoft investono in modo significativo nella concessione di licenze per i set di dati per migliorare i loro sistemi di visione di ML. Tuttavia, il fattore costo limita l'accessibilità per i ricercatori di aziende più piccole con budget limitati.

Il dominio legale della concessione di licenze per i dati ha visto casi importanti che aiutano a definire i parametri di utilizzo corretto. Un esempio importante è *Authors Guild, Inc. contro Google, Inc.* Questa causa del 2005 sosteneva che il progetto di scansione di libri di Google violava i diritti d'autore visualizzando frammenti senza autorizzazione. Tuttavia, i tribunali alla fine si sono pronunciati a favore di Google, sostenendo il "fair use" [correttezza] in base alla natura trasformativa della creazione di un indice ricercabile e della visualizzazione di estratti limitati di testo. Questo precedente fornisce alcune basi legali per sostenere che le protezioni del "fair use" si applicano all'indicizzazione di set di dati e alla generazione di campioni rappresentativi per l'apprendimento automatico. Tuttavia, le restrizioni di licenza rimangono vincolanti, quindi un'analisi completa dei termini di licenza è fondamentale. Il caso dimostra perché le negoziazioni con i fornitori di dati sono importanti per consentire un utilizzo legale entro limiti accettabili.

Nuove Normative sui Dati e le Loro Implicazioni

Anche le nuove normative sui dati hanno un impatto sulle pratiche di licenza. Il panorama legislativo si sta evolvendo con normative come l'Artificial Intelligence Act dell'UE, che è pronto a regolamentare lo sviluppo e l'uso dei sistemi di intelligenza artificiale all'interno dell'Unione Europea (UE). Questa legislazione:

1. Classifica i sistemi di IA in base al rischio.
2. Impone prerequisiti di sviluppo e utilizzo.
3. Sottolinea la qualità dei dati, la trasparenza, la supervisione umana e la responsabilità.

Inoltre, l'EU Act affronta le dimensioni etiche e le sfide operative in settori quali sanità e finanza. Gli elementi chiave includono il divieto di sistemi di intelligenza artificiale che presentano rischi "inaccettabili", condizioni rigorose per sistemi ad alto rischio e obblighi minimi per sistemi di intelligenza artificiale a "rischio limitato". Il proposto "European AI Board" supervisionerà e garantirà l'implementazione di una regolamentazione efficiente.

Problemi nell'Assemblaggio di Dataset di Training ML

Problemi complessi di licenza relativi a dati proprietari, leggi sul copyright e normative sulla privacy limitano le opzioni per l'assemblaggio dei set di dati di training ML. Tuttavia, espandere l'accessibilità tramite licenze più aperte o collaborazioni di dati pubblico-private potrebbe accelerare notevolmente il progresso del settore e gli standard etici.

A volte, alcune parti di un dataset potrebbero dover essere rimosse o oscurate per rispettare gli accordi di utilizzo dei dati o proteggere informazioni sensibili. Ad esempio, un set di dati di informazioni utente potrebbe contenere nomi, dettagli di contatto e altri dati identificativi che potrebbero dover essere rimossi dal set di dati; questo avviene molto tempo dopo che il set di dati è già stato attivamente reperito e utilizzato per l'addestramento dei modelli.

Analogamente, un dataset che include contenuti protetti da copyright o segreti commerciali potrebbe dover filtrare tali parti prima di essere distribuito. Leggi come il General Data Protection Regulation (GDPR), il California Consumer Privacy Act (CCPA) e L'Amended Act on the Protection of Personal Information (APPI) sono state approvate per garantire il diritto all'oblio. Queste normative impongono legalmente ai fornitori di modelli di cancellare i dati degli utenti su richiesta.

I raccoglitori e i fornitori di dati devono essere in grado di adottare misure appropriate per de-identificare o filtrare qualsiasi informazione proprietaria, concessa in licenza, riservata o regolamentata, se necessario. A volte, gli utenti possono richiedere esplicitamente che i loro dati vengano rimossi.

La possibilità di aggiornare il set di dati rimuovendo i dati consentirà ai creatori di rispettare gli obblighi legali ed etici relativi al loro utilizzo e alla privacy. Tuttavia, la capacità di rimuovere i dati presenta alcune limitazioni importanti. Dobbiamo considerare che alcuni

modelli potrebbero essere già stati addestrati sul dataset e non esiste un modo chiaro o noto per eliminare l'effetto di un particolare campione di dati dalla rete addestrata. Non esiste un meccanismo di cancellazione. Quindi, ciò solleva la questione: il modello dovrebbe essere riaddestrato da zero ogni volta che viene rimosso un campione? Questa è un'opzione costosa. Una volta che i dati sono stati utilizzati per addestrare un modello, la semplice rimozione dal set di dati originale potrebbe non eliminare completamente il suo impatto sul comportamento del modello. Sono necessarie nuove ricerche sugli effetti della rimozione dei dati sui modelli già addestrati e se sia necessario un ri-addestramento completo per evitare di conservare artefatti di dati eliminati. Ciò presenta una considerazione importante quando si bilanciano gli obblighi di licenza dei dati con l'efficienza e la praticità in un sistema di ML in evoluzione e distribuito.
La licenza del dataset è un dominio poliedrico che interseca tecnologia, etica e legge. Comprendere queste complessità diventa fondamentale per chiunque crei set di dati durante l'ingegneria dei dati, man mano che il mondo si evolve.

5.11 Conclusione

I dati sono il componente fondamentale dei sistemi di intelligenza artificiale. Senza dati di qualità, anche gli algoritmi di apprendimento automatico più avanzati falliranno. L'ingegneria dei dati comprende il processo end-to-end di raccolta, archiviazione, elaborazione e gestione dei dati per alimentare lo sviluppo di modelli di apprendimento automatico. Si inizia con la definizione chiara del problema principale e degli obiettivi, che guidano una raccolta dati efficace. I dati possono essere reperiti da diversi mezzi, tra cui dataset esistenti, web scraping, crowdsourcing e generazione di dati sintetici. Ogni approccio comporta compromessi tra costi, velocità, privacy e specificità. Una volta raccolti i dati, un'etichettatura ponderata tramite annotazione manuale o assistita dall'intelligenza artificiale consente la creazione di set di dati di training di alta qualità. Un'archiviazione adeguata in database, "warehouse" o "lake" facilita l'accesso e l'analisi. I metadati forniscono dettagli contestuali sui dati. L'elaborazione dei dati trasforma i dati grezzi in un formato pulito e coerente per lo sviluppo di modelli di apprendimento automatico. In tutta questa pipeline, la trasparenza attraverso la documentazione e il tracciamento della provenienza è fondamentale per l'etica, la verificabilità e la riproducibilità. I protocolli di licenza dei dati regolano anche l'accesso e l'uso legale dei dati. Le principali sfide nell'ingegneria dei dati includono rischi per la privacy, lacune di rappresentazione, restrizioni legali sui dati proprietari e la necessità di bilanciare vincoli concorrenti come velocità e qualità. Progettando attentamente dati di training di alta qualità, i professionisti dell'apprendimento automatico possono sviluppare sistemi di intelligenza artificiale accurati, robusti e responsabili, tra cui applicazioni embedded e TinyML.

5.12 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

ⓘ Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Data Engineering: Overview.
- Feature engineering.
- Data Standards: Speech Commands.
- Crowdsourcing Data for the Long Tail.
- Reusing and Adapting Existing Datasets.
- Responsible Data Collection.
- Rilevamento Dati Anomali:
  - Anomaly Detection: Overview.
  - Anomaly Detection: Challenges.
  - Anomaly Detection: Datasets.
  - Anomaly Detection: Using Autoencoders.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 5.1
- Esercizio 5.2
- Esercizio 5.3
- Esercizio 5.4
- Esercizio 5.5

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

6. Framework di IA

DALL·E 3 Prompt: Illustrazione in formato rettangolare, progettata per un libro di testo professionale, in cui il contenuto si estende per l'intera larghezza. Il grafico vivace rappresenta i framework di training e inferenza per l'ML. Le icone per TensorFlow, Keras, PyTorch, ONNX e TensorRT sono distribuite, riempiono l'intero spazio orizzontale e sono allineate verticalmente. Ogni icona è accompagnata da brevi annotazioni che ne descrivono le caratteristiche. I colori vividi come blu, verde e arancione evidenziano le icone e le sezioni su uno sfondo sfumato più morbido. La distinzione tra framework di training e inferenza è accentuata tramite sezioni codificate a colori, con linee pulite e tipografia moderna che mantengono chiarezza e attenzione.

Questo capitolo esplora il panorama dei framework di intelligenza artificiale che fungono da base per lo sviluppo di sistemi di apprendimento automatico. I framework di intelligenza artificiale forniscono gli strumenti, le librerie e gli ambienti per progettare, addestrare e distribuire modelli di apprendimento automatico. Esploreremo il percorso evolutivo di questi framework, analizzeremo il funzionamento di TensorFlow e forniremo approfondimenti sui componenti principali e sulle funzionalità avanzate che li definiscono.
Inoltre, esaminiamo la specializzazione dei framework su misura per esigenze specifiche, l'emergere di framework progettati espressamente per l'intelligenza artificiale embedded e i criteri per selezionare il framework più adatto ai vari progetti. Questa esplorazione sarà completata da uno sguardo alle tendenze future che dovrebbero modellare il panorama dei framework di apprendimento automatico nei prossimi anni.

Obiettivi dell'Apprendimento

- Comprendere l'evoluzione e le capacità dei principali framework di apprendimento automatico. Ciò include modelli di esecuzione di grafici, paradigmi di programmazione, supporto per l'accelerazione hardware e come si sono espansi nel tempo.
- Scoprire i componenti principali e le funzionalità dei framework, come grafi computazionali, pipeline di dati, algoritmi di ottimizzazione, loop di training, ecc., che consentono una creazione di modelli efficiente.
- Confrontare i framework in diversi ambienti, come cloud, edge e TinyML. Scoprire come i framework si specializzano in base a vincoli computazionali e hardware.
- Approfondire i framework embedded e focalizzarsi su TinyML come TensorFlow Lite Micro, CMSIS-NN, TinyEngine, ecc., e come ottimizzano per i microcontrollori.
- Quando si sceglie un framework, si esplorano le considerazioni sulla conversione e l'implementazione del modello, tra cui latenza, utilizzo della memoria e supporto hardware.
- Valutare i fattori chiave nella selezione del framework giusto, come prestazioni, compatibilità hardware, supporto della community, facilità d'uso, ecc., in base alle esigenze e ai vincoli specifici del progetto.
- Comprendere i limiti dei framework attuali e le potenziali tendenze future, come l'uso del ML per migliorare i framework, i sistemi ML decomposti e i compilatori ad alte prestazioni.

6.1 Introduzione

I framework di machine learning [apprendimento automatico] forniscono gli strumenti e l'infrastruttura per creare, addestrare e distribuire in modo efficiente modelli di apprendimento automatico. In questo capitolo esploreremo l'evoluzione e le capacità chiave dei principali framework come TensorFlow (TF), PyTorch e framework specializzati per dispositivi embedded. Ci immergeremo nei componenti come grafi computazionali, algoritmi di ottimizzazione, accelerazione hardware e altro che consentono agli sviluppatori di creare rapidamente modelli performanti. Comprendere questi framework è essenziale per sfruttare la potenza del deep learning in tutto lo spettro, dal cloud ai dispositivi edge [periferici].

I framework di apprendimento automatico gestiscono gran parte della complessità dello sviluppo di modelli tramite API di alto livello e linguaggi specifici per dominio che consentono ai professionisti di creare rapidamente modelli combinando componenti e astrazioni predefiniti. Ad esempio, framework come TensorFlow e PyTorch forniscono API Python per definire architetture di reti neurali utilizzando livelli, ottimizzatori, set di dati e altro. Ciò consente un'iterazione rapida rispetto alla codifica di ogni dettaglio del modello partendo da zero.

Una capacità chiave offerta da questi framework è rappresentata dai motori di training distribuiti che possono scalare l'addestramento del modello su cluster di GPU e TPU. Ciò rende possibile il training di modelli all'avanguardia con miliardi o trilioni di parametri su vasti set di dati. I framework si integrano anche con hardware specializzato come le GPU NVIDIA per accelerare ulteriormente il training tramite ottimizzazioni come la parallelizzazione ed efficienti operazioni matriciali.

Inoltre, i framework semplificano il deploy distribuzione di modelli finiti in produzione tramite strumenti come TensorFlow Serving per il model serving scalabile e TensorFlow Lite per l'ottimizzazione su dispositivi mobili ed edge. Altre capacità preziose includono visualizzazione, tecniche di ottimizzazione del modello come quantizzazione e potatura e monitoraggio delle metriche durante il training.

I principali framework open source come TensorFlow, PyTorch e MXNet alimentano gran parte della ricerca e dello sviluppo dell'IA oggi. Offerte commerciali come Amazon SageMaker e Microsoft Azure Machine Learning integrano questi framework open source con funzionalità proprietarie e strumenti aziendali.
Gli ingegneri e i professionisti del machine learning sfruttano questi framework robusti per concentrarsi su attività di alto valore come architettura del modello, progettazione delle feature e ottimizzazione degli iperparametri anziché sull'infrastruttura. L'obiettivo è creare e distribuire modelli performanti che risolvano in modo efficiente i problemi del mondo reale.
In questo capitolo, esploreremo i principali framework cloud odierni e il modo in cui hanno adattato modelli e strumenti specificamente per la distribuzione embedded ed edge. Confronteremo modelli di programmazione, hardware supportato, capacità di ottimizzazione e altro ancora per comprendere appieno in che modo i framework consentono un apprendimento automatico scalabile dal cloud all'edge.

6.2 Evoluzione dei Framework

I framework di apprendimento automatico si sono evoluti in modo significativo per soddisfare le diverse esigenze dei professionisti del machine learning e i progressi nelle tecniche di intelligenza artificiale. Qualche decennio fa, la creazione e l'addestramento di modelli di apprendimento automatico richiedevano un'ampia codifica e infrastruttura di basso livello. Oltre alla necessità di una codifica di basso livello, la prima ricerca sulle reti neurali era limitata da dati e potenza di calcolo insufficienti. Tuttavia, i framework di apprendimento automatico si sono evoluti notevolmente nell'ultimo decennio per soddisfare le crescenti esigenze dei professionisti e i rapidi progressi nelle tecniche di deep learning [apprendimento profondo]. Il rilascio di grandi set di dati come ImageNet (Deng et al. 2009) e i progressi nel calcolo parallelo con GPU hanno sbloccato il potenziale per reti neurali molto più profonde.
I primi framework di apprendimento automatico, Theano di Team et al. (2016) e Caffe di Y. Jia et al. (2014), sono stati sviluppati da istituzioni accademiche. Theano è stato creato dal Montreal Institute for Learning Algorithms, mentre Caffe è stato sviluppato dal Berkeley Vision and Learning Center. Nel crescente interesse per il deep learning dovuto alle prestazioni all'avanguardia di AlexNet Krizhevsky, Sutskever, e Hinton (2012) sul dataset ImageNet, aziende private e singole persone hanno iniziato a sviluppare framework di ML, dando vita a Keras di Chollet (2018), Chainer di Tokui et al. (2019), TensorFlow di Google (Yu et al. 2018), CNTK di Microsoft (Seide e Agarwal 2016) e PyTorch di Facebook (Ansel et al. 2024).
Molti di questi framework ML possono essere suddivisi in framework di alto livello, di basso livello e di grafi computazionali statici e dinamici. I framework di alto livello forniscono un livello di astrazione più elevato rispetto a quelli di basso livello. I framework di alto livello hanno funzioni e moduli predefiniti per attività ML comuni, come la creazione, l'addestramento e la valutazione di modelli ML comuni, la preelaborazione dei dati, le funzionalità di progettazione e la visualizzazione dei dati, che i framework di basso livello non hanno. Pertanto, i framework di alto livello possono risultare più facili da usare ma sono meno personalizzabili rispetto a quelli di basso livello (ad esempio, gli utenti di framework di basso livello possono definire livelli personalizzati, funzioni "loss" [di perdita], algoritmi di ottimizzazione, ecc.). Esempi di framework di alto livello sono TensorFlow/Keras e PyTorch. Esempi di framework ML di basso livello includono TensorFlow con API di basso livello, Theano, Caffe, Chainer e CNTK.
Framework come Theano e Caffe utilizzavano grafi computazionali statici, che richiedevano la definizione anticipata dell'architettura completa del modello, limitandone così la flessibilità. Al contrario, i grafici dinamici vengono costruiti al volo per uno sviluppo più iterativo. Intorno al 2016, framework come PyTorch e TensorFlow 2.0 hanno iniziato ad

adottare grafici dinamici, offrendo maggiore flessibilità per lo sviluppo del modello. Discuteremo di questi concetti e dettagli più avanti nella sezione Training dell'IA.
Lo sviluppo di questi framework ha suscitato un'esplosione di dimensioni e complessità del modello nel tempo, dai primi perceptron multistrato e reti convoluzionali ai moderni trasformatori con miliardi o trilioni di parametri. Nel 2016, i modelli ResNet di K. He et al. (2016) hanno raggiunto un'accuratezza ImageNet record con oltre 150 livelli e 25 milioni di parametri. Poi, nel 2020, il modello linguistico GPT-3 di OpenAI (Brown et al. 2020) ha spinto i parametri a un sorprendente numero di 175 miliardi utilizzando il parallelismo del modello nei framework per addestrare migliaia di GPU e TPU.
Ogni generazione di framework ha sbloccato nuove capacità che hanno alimentato il progresso:

- Theano e TensorFlow (2015) hanno introdotto grafi computazionali e differenziazione automatica per semplificare la creazione di modelli.
- CNTK (2016) ha aperto la strada a un addestramento distribuito efficiente combinando parallelismo di modelli e dati.
- PyTorch (2016) ha fornito programmazione imperativa e grafici dinamici per una sperimentazione flessibile.
- TensorFlow 2.0 (2019) ha impostato di default l'esecuzione Eager per intuitività e debug.
- TensorFlow Graphics (2020) ha aggiunto strutture dati 3D per gestire nuvole di punti e mesh.

Negli ultimi anni, il panorama dei framework di apprendimento automatico si è notevolmente consolidato. Figura 6.2 illustra questa convergenza, mostrando che TensorFlow e PyTorch sono diventati i framework ML prevalentemente dominanti, rappresentando collettivamente oltre il 95% dei framework ML utilizzati nella ricerca e nella produzione. Sebbene entrambi i framework siano diventati importanti, presentano caratteristiche distinte. Figura 6.1 traccia un contrasto tra gli attributi di TensorFlow e PyTorch, contribuendo a spiegare il loro predominio complementare nel settore.

TensorFlow	PyTorch
Written in C++ and is, as a result, very fast and efficient.	Written in Python making it more accessible and flattening the learning curve. However, the C++ core means PyTorch is still quite fast.
Feature rich; TensorFlow can be used for training data as well as for inference.	Very flexible; as data size can also be changed during data training.
Very good documentation; TensorFlow has many users and an big community which has led to strong documentation.	Popular at research level; Pytorch was by far the most talked about ML library at CVPR, one of the most important computer vision conferences.
High popularity; TensorFlow has established itself as the most used ML library over a number of years now.	Rapid growth in popularity in both business and research use cases.
Many APIs available; TensorFlow is a library with a rich choice of easy to use APIs.	Many libraries available; PyTorch is composed of multiple libraries and platforms.
Supports JavaScript; TensorFlow supports JavaScript, C++ and Java in addition to Python.	Python-based; PyTorch allows developers to write code in Python
For Mobile & IoT, inferences can be performed with TensorFlow Lite on mobile devices such as Android or iOS, as well as on Edge TPU or Raspberry Pi.	PyTorch API; the PyTorch API is often preferred as it is better designed - plus TensorFlow has historically changed their API frequently.

Figura 6.1: PyTorch e TensorFlow: Caratteristiche e Funzioni. Fonte: K&C

Figura 6.2: Popolarità dei framework ML negli Stati Uniti misurata dalle ricerche web di Google. Fonte: Google.

Un approccio unico non funziona bene in tutto lo spettro, dal cloud ai piccoli dispositivi edge. Diversi framework rappresentano varie filosofie sull'esecuzione di grafici, API dichiarative rispetto a quelle imperative e altro ancora. Le dichiarative definiscono cosa dovrebbe fare il programma, mentre le imperative si concentrano su come dovrebbe essere fatto passo dopo passo. Ad esempio, TensorFlow utilizza l'esecuzione di grafici e la modellazione in stile dichiarativo, mentre PyTorch adotta l'esecuzione rapida e la modellazione imperativa per una maggiore flessibilità con Python. Ogni approccio comporta dei compromessi che discuteremo in Sezione 6.3.7.

Gli attuali framework avanzati consentono ai professionisti di sviluppare e distribuire modelli sempre più complessi, un fattore chiave dell'innovazione nel campo dell'intelligenza artificiale. Questi framework continuano a evolversi ed espandere le loro capacità per la prossima generazione di machine learning. Per capire come questi sistemi continuano a evolversi, approfondiremo TensorFlow come esempio di come il framework sia cresciuto in complessità nel tempo.

6.3 Approfondimento su TensorFlow

TensorFlow è stato sviluppato dal team di Google Brain ed è stato rilasciato come libreria software open source il 9 novembre 2015. È stato progettato per il calcolo numerico utilizzando grafici di flusso di dati e da allora è diventato popolare per un'ampia gamma di applicazioni di apprendimento automatico e deep learning.

TensorFlow è un framework di training e inferenza che fornisce funzionalità integrate per gestire tutto, dalla creazione e training del modello alla distribuzione, come mostrato in Figura 6.3. Sin dal suo sviluppo iniziale, l'ecosistema TensorFlow è cresciuto fino a includere molte diverse "varietà" di TensorFlow, ciascuna pensata per consentire agli utenti di supportare ML su diverse piattaforme. In questa sezione, discuteremo principalmente solo del pacchetto core.

6.3.1 Ecosistema TF

1. TensorFlow Core: pacchetto principale con cui interagiscono la maggior parte degli sviluppatori. Fornisce una piattaforma completa e flessibile per definire, addestrare e distribuire modelli di apprendimento automatico. Include tf.keras come API di alto livello.
2. TensorFlow Lite: progettato per distribuire modelli leggeri su dispositivi mobili, embedded ed edge. Offre strumenti per convertire i modelli TensorFlow in un formato più compatto adatto a dispositivi con risorse limitate e fornisce modelli pre-addestrati ottimizzati per dispositivi mobili.
3. TensorFlow Lite Micro: progettato per eseguire modelli di apprendimento automatico su microcontrollori con risorse minime. Funziona senza la necessità di supporto del sistema operativo, librerie C o C++ standard o allocazione dinamica della memoria, utilizzando solo pochi kilobyte di memoria.
4. TensorFlow.js: libreria JavaScript che consente l'addestramento e la distribuzione di modelli di apprendimento automatico direttamente nel browser o su Node.js. Fornisce inoltre strumenti per il porting di modelli TensorFlow pre-addestrati nel formato browser-friendly.
5. TensorFlow su dispositivi edge (Coral): piattaforma di componenti hardware e strumenti software di Google che consente l'esecuzione di modelli TensorFlow su dispositivi edge, sfruttando Edge TPU per l'accelerazione.
6. TensorFlow Federated (TFF): framework per l'apprendimento automatico e altri calcoli su dati decentralizzati. TFF facilita l'apprendimento "federato", consentendo l'addestramento del modello su molti dispositivi senza centralizzare i dati.
7. TensorFlow Graphics: libreria per l'utilizzo di TensorFlow per svolgere attività correlate alla grafica, tra cui l'elaborazione di forme 3D e nuvole di punti, utilizzando il deep learning.
8. TensorFlow Hub: repository di componenti di modelli di apprendimento automatico riutilizzabili che consente agli sviluppatori di riutilizzare componenti di modelli pre-addestrati, facilitando l'apprendimento per trasferimento e la composizione del modello.
9. TensorFlow Serving: framework progettato per servire e distribuire modelli di apprendimento automatico per l'inferenza in ambienti di produzione. Fornisce strumenti per il versioning e l'aggiornamento dinamico dei modelli distribuiti senza interruzione del servizio.
10. TensorFlow Extended (TFX): piattaforma end-to-end progettata per distribuire e gestire pipeline di apprendimento automatico in ambienti di produzione. TFX comprende validazione dei dati, pre-elaborazione, addestramento del modello, convalida e componenti di servizio.

Figura 6.3: Panoramica dell'architettura di TensorFlow 2.0. Fonte: Tensorflow.

TensorFlow è stato sviluppato per affrontare le limitazioni di DistBelief (Yu et al. 2018)—il framework in uso presso Google dal 2011 al 2015—offrendo flessibilità lungo tre direttrici: 1) definizione di nuovi livelli [livelli], 2) perfezionamento degli algoritmi di training e 3) definizione di nuovi algoritmi di training. Per comprendere quali limitazioni di DistBelief hanno portato allo sviluppo di TensorFlow, faremo prima una breve panoramica dell'architettura del server dei parametri utilizzata da DistBelief (Dean et al. 2012).
L'architettura Parameter Server (PS) è un design popolare per distribuire il training di modelli di apprendimento automatico, in particolare reti neurali profonde, su più macchine. L'idea fondamentale è di separare l'archiviazione e la gestione dei parametri del modello dal calcolo utilizzato per aggiornare tali parametri. In genere, i server dei parametri gestiscono l'archiviazione e la gestione dei parametri del modello, suddividendoli su più server. I processi worker eseguono le attività di calcolo, tra cui l'elaborazione dei dati e il calcolo dei gradienti, che vengono poi inviati ai server dei parametri per l'aggiornamento.

Storage: I processi del server dei parametri stateful [con stato] gestivano l'archiviazione e la gestione dei parametri del modello. Data l'ampia scala dei modelli e la natura distribuita del sistema, questi parametri erano condivisi tra più server dei parametri. Ogni server manteneva una parte dei parametri del modello, rendendolo "stateful" poiché doveva mantenere e gestire questo stato durante il processo di training.

Computation: I processi worker, che potevano essere eseguiti in parallelo, erano senza stato e puramente computazionali. Elaboravano dati e calcolavano gradienti senza mantenere alcuno stato o memoria a lungo termine (M. Li et al. 2014). I worker non conservavano informazioni tra le diverse attività. Invece, comunicavano periodicamente con i server dei parametri per recuperare i parametri più recenti e restituire i gradienti calcolati.

Esercizio 6.1: TensorFlow Core

Andiamo a comprendere in modo completo gli algoritmi di apprendimento automatico di base utilizzando TensorFlow e le loro applicazioni pratiche nell'analisi dei dati e nella modellazione predittiva. Inizieremo con la regressione lineare per prevedere i tassi di sopravvivenza dal set di dati del Titanic. Poi, utilizzando TensorFlow, costruiremo classificatori per identificare diverse specie di fiori in base ai loro attributi. Successivamente, utilizzeremo l'algoritmo K-Means e la sua applicazione nella segmentazione dei set di dati in cluster coesi. Infine, applicheremo modelli hidden [nascosti] di Markov (HMM) per prevedere i pattern meteorologici.

Open in Colab

Esercizio 6.2: TensorFlow Lite

Qui vedremo come costruire un modello di apprendimento automatico in miniatura per microcontrollori. Costruiremo una mini rete neurale semplificata per apprendere dai dati anche con risorse limitate e ottimizzata per l'implementazione riducendo il nostro modello per un uso efficiente sui microcontrollori. TensorFlow Lite, una potente tecnologia derivata da TensorFlow, riduce i modelli per dispositivi minuscoli e aiuta ad abilitare funzionalità sul dispositivo come il riconoscimento delle immagini nei dispositivi smart [intelligenti]. Viene utilizzato nell'edge computing per consentire analisi e decisioni più rapide nei dispositivi che elaborano i dati localmente.

Open in Colab

DistBelief e la sua architettura definita sopra sono stati fondamentali per abilitare il deep learning distribuito in Google, ma hanno anche introdotto delle limitazioni che hanno motivato lo sviluppo di TensorFlow:

6.3.2 Grafico di Calcolo Statico

I parametri del modello sono distribuiti su vari server di parametri nell'architettura del server di parametri. Poiché DistBelief è stato progettato principalmente per il paradigma della rete neurale, i parametri corrispondevano a una struttura di rete neurale fissa. Se il computation graph [grafico di calcolo] fosse dinamico, la distribuzione e il coordinamento dei parametri diventerebbero significativamente più complicati. Ad esempio, una modifica nel grafico potrebbe richiedere l'inizializzazione di nuovi parametri o la rimozione di quelli esistenti, complicando le attività di gestione e sincronizzazione dei server di parametri. Ciò ha reso più difficile implementare modelli al di fuori del framework neurale o modelli che richiedevano grafici di calcolo dinamici.
TensorFlow è stato progettato come un framework di calcolo più generale che esprime il calcolo come un grafico del flusso di dati. Ciò consente una più ampia varietà di modelli e algoritmi di apprendimento automatico al di fuori delle reti neurali e fornisce flessibilità nel perfezionamento dei modelli.

6.3.3 Usabilità & Distribuzione

Il modello del server dei parametri delinea i ruoli (nodi worker e server dei parametri) ed è ottimizzato per i deployment [distribuzioni] dei data center, che potrebbero essere ottimali solo per alcuni casi d'uso. Ad esempio, questa divisione introduce overhead o complessità sui dispositivi edge o in altri ambienti non data center.
TensorFlow è stato creato per funzionare su più piattaforme, dai dispositivi mobili e edge all'infrastruttura cloud. Mirava anche a essere più leggero e intuitivo per gli sviluppatori e a fornire facilità d'uso tra il training locale e quello distribuito.

6.3.4 Progettazione dell'Architettura

Invece di utilizzare l'architettura del server dei parametri, TensorFlow distribuisce i task [attività] su un cluster. Queste attività sono processi denominati che possono comunicare su una rete e ciascuna può eseguire la struttura principale di TensorFlow, il grafico del flusso di dati e l'interfaccia con vari dispositivi di elaborazione (come CPU o GPU). Questo grafico [grafo] è una rappresentazione diretta in cui i nodi simboleggiano le operazioni di elaborazione e gli edge rappresentano i tensori (dati) che scorrono tra queste operazioni. Nonostante l'assenza di server di parametri tradizionali, alcuni "task PS" memorizzano e gestiscono parametri che ricordano i server di parametri di altri sistemi. I task rimanenti, che di solito gestiscono calcoli, elaborazione dati e gradienti, sono denominati "task worker". I task PS di TensorFlow possono eseguire qualsiasi calcolo rappresentabile dal grafico del flusso di dati, il che significa che non sono limitati solo all'archiviazione dei parametri e il calcolo può essere distribuito. Questa capacità li rende significativamente più versatili e offre agli utenti il potere di programmare i task PS utilizzando l'interfaccia TensorFlow standard, la stessa che userebbero per definire i loro modelli. Come accennato in precedenza, la struttura dei grafici del flusso di dati li rende anche intrinsecamente buoni per il parallelismo, consentendo l'elaborazione di grandi set di dati.

6.3.5 Funzionalità Native & Keras

TensorFlow include librerie per aiutare gli utenti a sviluppare e distribuire più modelli specifici per i casi d'uso e, poiché questo framework è open source, questo elenco continua a crescere. Queste librerie affrontano l'intero ciclo di vita dello sviluppo ML: preparazione dei dati, creazione di modelli, distribuzione e IA responsabile.
Uno dei maggiori vantaggi di TensorFlow è la sua integrazione con Keras, anche se, come vedremo nella prossima sezione, Pytorch ha recentemente aggiunto un'integrazione Keras. Keras è un altro framework ML creato per essere estremamente intuitivo e, di conseguenza, ha un alto livello di astrazione. Parleremo di Keras più approfonditamente più avanti in questo capitolo. Tuttavia, quando si discute della sua integrazione con TensorFlow, è importante notare che era stato originariamente creato per essere indipendente dal backend. Ciò significa che gli utenti potrebbero astrarre queste complessità, offrendo un modo più pulito e intuitivo per definire e addestrare modelli senza preoccuparsi di problemi di compatibilità con diversi backend. Gli utenti di TensorFlow hanno evidenziato alcuni problemi sull'usabilità e la leggibilità dell'API di TensorFlow, quindi, man mano che TF acquisiva importanza, ha integrato Keras come API di alto livello. Questa integrazione ha offerto grandi vantaggi agli utenti di TensorFlow poiché ha introdotto una leggibilità e una portabilità più intuitive dei modelli, sfruttando comunque le potenti funzionalità di backend, il supporto di Google e l'infrastruttura per distribuire i modelli su varie piattaforme.

Esercizio 6.3: Esplorazione di Keras: Creazione, Addestramento e Valutazione di Reti Neurali

Qui, impareremo come utilizzare Keras, un'API di reti neurali di alto livello, per lo sviluppo e l'addestramento (training) di modelli. Esploreremo l'API funzionale per la creazione di modelli concisi, comprenderemo le classi "loss" e metriche per la valutazione dei modelli e utilizzeremo gli ottimizzatori nativi per aggiornare i parametri del modello durante l'addestramento. Inoltre, scopriremo come definire layer e metriche personalizzati su misura per le nostre esigenze. Infine, esamineremo i cicli di addestramento di Keras per semplificare il processo di addestramento delle reti neurali su grandi set di dati. Questa conoscenza ci consentirà di costruire e ottimizzare modelli di reti neurali in varie applicazioni di machine learning e intelligenza artificiale.

Open in Colab

6.3.6 Limitazioni e Sfide

TensorFlow è uno dei framework di deep learning più popolari, ma ha dovuto affrontare critiche e debolezze, principalmente legate all'usabilità e all'utilizzo delle risorse. Sebbene vantaggioso, il ritmo rapido degli aggiornamenti tramite il supporto di Google ha talvolta portato a problemi di retrocompatibilità, funzioni deprecate e documentazione instabile. Inoltre, anche con l'implementazione di Keras, la sintassi e la curva di apprendimento di TensorFlow possono risultare difficili per i nuovi utenti. Un'altra critica importante di TensorFlow è il suo elevato overhead e consumo di memoria dovuto alla gamma di librerie integrate e al supporto. Sebbene le versioni ridotte possano risolvere alcuni di questi problemi, potrebbero comunque essere limitate in ambienti con risorse limitate.

6.3.7 PyTorch & TensorFlow

PyTorch e TensorFlow si sono affermati come leader nel settore. Entrambi i framework offrono funzionalità robuste ma differiscono per filosofie di progettazione, facilità d'uso, ecosistema e capacità di distribuzione.

Filosofia di Progettazione e Paradigma di Programmazione: PyTorch utilizza un grafo computazionale dinamico denominato eager execution [esecuzione rapida]. Ciò lo rende intuitivo e facilita il debug poiché le operazioni vengono eseguite immediatamente e possono essere ispezionate al volo. Al contrario, le versioni precedenti di TensorFlow erano incentrate su un grafo computazionale statico, che richiedeva la definizione completa del grafico prima dell'esecuzione. Tuttavia, TensorFlow 2.0 ha introdotto la "eager execution" per default, rendendolo più allineato con PyTorch. La natura dinamica di PyTorch e l'approccio basato su Python hanno consentito la sua semplicità e flessibilità, in particolare per la prototipazione rapida. L'approccio grafico statico di TensorFlow nelle sue versioni precedenti aveva una curva di apprendimento più ripida; l'introduzione di TensorFlow 2.0, con la sua integrazione Keras come API di alto livello, ha semplificato notevolmente il processo di sviluppo.

Deployment: PyTorch è fortemente favorito negli ambienti di ricerca, ma la distribuzione dei modelli PyTorch in contesti di produzione è sempre stata un problema. Tuttavia, la distribuzione è diventata più fattibile con l'introduzione di TorchScript, lo strumento TorchServe e PyTorch Mobile. TensorFlow si distingue per la sua forte scalabilità e capacità di distribuzione, in particolare su piattaforme embedded e mobili con TensorFlow Lite. TensorFlow Serving e TensorFlow.js facilitano ulteriormente la distribuzione in vari ambienti, conferendogli così una portata più ampia nell'ecosistema.

Prestazioni: Entrambi i framework offrono un'accelerazione hardware efficiente per le loro operazioni. Tuttavia, TensorFlow ha un flusso di lavoro di ottimizzazione leggermente più robusto, come il compilatore XLA (Accelerated Linear Algebra), che può aumentare

ulteriormente le prestazioni. Il suo grafo computazionale statico era anche vantaggioso per alcune ottimizzazioni nelle prime versioni.

Ecosistema: PyTorch ha un ecosistema in crescita con strumenti come TorchServe per servire modelli e librerie come TorchVision, TorchText e TorchAudio per domini specifici. Come abbiamo detto prima, TensorFlow ha un ecosistema ampio e maturo. TensorFlow Extended (TFX) fornisce una piattaforma end-to-end per distribuire pipeline di apprendimento automatico di produzione. Altri strumenti e librerie includono TensorFlow Lite, TensorFlow Lite Micro, TensorFlow.js, TensorFlow Hub e TensorFlow Serving. Tabella 6.1 fornisce un'analisi comparativa:

Tabella 6.1: Confronto tra PyTorch e TensorFlow.

Aspetto	Pytorch	TensorFlow
Filosofia di Progettazione	Grafo computazionale dinamico (eager execution)	Grafo computazionale statico (prime versioni); Esecuzione rapida in TensorFlow 2.0
Deployment	Tradizionalmente impegnativa; Migliorata con TorchScript e TorchServe	Scalabile, specialmente su piattaforme embedded con TensorFlow Lite
Prestazioni e Ottimizzazione	Accelerazione GPU efficiente	Ottimizzazione robusta con compilatore XLA
Ecosistema	TorchServe, TorchVision, TorchText, TorchAudio, PyTorch Mobile	TensorFlow Extended (TFX), TensorFlow Lite, TensorFlow Lite Micro TensorFlow.js, TensorFlow Hub, TensorFlow Serving
Facilità d'uso	Preferito per il suo approccio Pythonic e la prototipazione rapida	Curva di apprendimento inizialmente ripida; Semplificato con Keras in TensorFlow 2.0

6.4 Componenti di Base del Framework

Dopo aver introdotto i popolari framework di machine learning e aver fornito un confronto di alto livello, questa sezione presenterà le funzionalità principali che formano la struttura di questi framework. Tratterà la struttura speciale chiamata tensori, che questi framework utilizzano per gestire più facilmente dati multidimensionali complessi. Si vedrà anche come questi framework rappresentano diversi tipi di architetture di reti neurali e le loro operazioni richieste tramite grafi computazionali. Inoltre, si vedrà come offrono strumenti che rendono lo sviluppo di modelli di machine learning più astratto ed efficiente, come caricatori di dati, algoritmi di ottimizzazione delle perdite implementate, tecniche di differenziazione efficienti e la capacità di accelerare il processo di training su acceleratori hardware.

6.4.1 Strutture Dati Tensoriali

Come mostrato nella figura, i vettori possono essere rappresentati come una pila di numeri in un array unidimensionale. Le matrici seguono la stessa idea e si possono pensare a loro come a molti vettori impilati l'uno sull'altro, rendendoli bidimensionali. I tensori di dimensioni superiori funzionano allo stesso modo. Un tensore tridimensionale, come illustrato in Figura 6.4, è semplicemente un set di matrici impilate l'una sull'altra in un'altra direzione. Pertanto, vettori e matrici possono essere considerati casi speciali di tensori con dimensioni 1D e 2D, rispettivamente.

Figura 6.4: Visualizzazione della Struttura Dati Tensore.

I tensori offrono una struttura flessibile che può rappresentare dati in dimensioni superiori. Figura 6.5 illustra come questo concetto si applica ai dati immagine. Come mostrato nella figura, le immagini non sono rappresentate da una sola matrice di valori pixel. Invece, hanno in genere tre canali, dove ogni canale è una matrice contenente valori pixel che rappresentano l'intensità di rosso, verde o blu. Insieme, questi canali creano un'immagine colorata. Senza i tensori, archiviare tutte queste informazioni da più matrici può risultare complesso. Tuttavia, come illustra Figura 6.5, i tensori facilitano il contenimento dei dati dell'immagine in un'unica struttura tridimensionale, in cui ciascun numero rappresenta un determinato valore di colore in una posizione specifica nell'immagine.

Figura 6.5: Visualizzazione della struttura dell'immagine colorata che può essere facilmente memorizzata come un Tensore 3D. Credito: Niklas Lang

Non finisce qui. Se volessimo archiviare una serie di immagini, potremmo usare un tensore quadridimensionale, in cui la nuova dimensione rappresenta immagini diverse. Ciò significa che si stanno archiviando più immagini, ciascuna con tre matrici che rappresentano i tre canali del colore. Questo dà un'idea dell'utilità dei tensori quando si gestiscono dati multidimensionali in modo efficiente.
I tensori hanno anche un attributo unico che consente ai framework di calcolare automaticamente i gradienti, semplificando l'implementazione di modelli complessi e algoritmi di ottimizzazione. Nel machine learning, come discusso nel Capitolo 3, la backpropagation richiede di prendere la derivata delle equazioni. Una delle caratteristiche principali dei tensori in PyTorch e TensorFlow è la loro capacità di tracciare i calcoli e calcolare i gradienti. Ciò è fondamentale per la backpropagation nelle reti neurali. Ad esempio, in PyTorch, si può usare l'attributo `requires_grad`, che consente di calcolare e memorizzare automaticamente i gradienti durante il "backward pass", facilitando il processo di ottimizzazione. Analogamente, in TensorFlow, `tf.GradientTape` registra le operazioni per la differenziazione automatica.
Si consideri questa semplice equazione matematica che si vuole differenziare.
Matematicamente, il calcolo del gradiente si effettua nel modo seguente:
Dato:

$$y = x^2$$

La derivata di y rispetto a x è:

$$\frac{dy}{dx} = 2x$$

Quando $x = 2$:

$$\frac{dy}{dx} = 2 * 2 = 4$$

Il gradiente di y rispetto a x, con $x = 2$, è 4.
Una potente caratteristica dei tensori in PyTorch e TensorFlow è la loro capacità di calcolare facilmente le derivate (gradienti). Ecco gli esempi di codice corrispondenti in PyTorch e TensorFlow:

PyTorch

```
import torch

# Create a tensor with gradient tracking
x = torch.tensor(2.0, requires_grad=True)

# Define a simple function
y = x ** 2

# Compute the gradient
y.backward()

# Print the gradient
print(x.grad)

# Output
tensor(4.0)
```

TensorFlow

```
import tensorflow as tf

# Create a tensor with gradient tracking
x = tf.Variable(2.0)

# Define a simple function
```

```
with tf.GradientTape() as tape:
    y = x ** 2

# Compute the gradient
grad = tape.gradient(y, x)

# Print the gradient
print(grad)

# Output
tf.Tensor(4.0, shape=(), dtype=float32)
```

Questa differenziazione automatica è una potente funzionalità dei tensori in framework come PyTorch e TensorFlow, che semplifica l'implementazione e l'ottimizzazione di modelli complessi di apprendimento automatico.

6.4.2 Grafi computazionali

6.4.2.1 Definizione di Grafico

I grafi computazionali sono una componente chiave di framework di deep learning come TensorFlow e PyTorch. Ci consentono di esprimere architetture di reti neurali complesse in modo efficiente e differenziato. Un grafo computazionale è costituito da un grafo aciclico diretto (directed acyclic graph, DAG) in cui ogni nodo rappresenta un'operazione o una variabile e gli spigoli rappresentano le dipendenze dei dati tra di essi.
È importante distinguere i grafici computazionali dai diagrammi delle reti neurali, come quelli dei "multilayer perceptrons (MLP)", che rappresentano nodi e layer. I diagrammi di reti neurali, come illustrato nel Capitolo 3, visualizzano l'architettura e il flusso di dati attraverso nodi e layer, fornendo una comprensione intuitiva della struttura del modello. Al contrario, i grafi computazionali forniscono una rappresentazione di basso livello delle operazioni matematiche sottostanti e delle dipendenze dei dati necessarie per implementare e addestrare queste reti.
Ad esempio, un nodo potrebbe rappresentare un'operazione di moltiplicazione di matrici, prendendo due matrici di input (o tensori) e producendo una matrice di output (o tensore). Per visualizzarlo, si consideri il semplice esempio in Figura 6.6. Il grafo aciclico orientato calcola $z = x \times y$, dove ogni variabile è solo un numero.

Figura 6.6: Esempio di base di un grafo computazionale.

Framework come TensorFlow e PyTorch creano grafi computazionali per implementare le architetture delle reti neurali che in genere rappresentiamo con diagrammi. Quando si definisce un layer di rete neurale nel codice (ad esempio, un "layer denso" in TensorFlow), il framework costruisce un grafo computazionale che include tutte le operazioni necessarie (come moltiplicazione di matrici, addizione e funzioni di attivazione) e le relative dipendenze dai dati. Questo grafo consente al framework di gestire in modo efficiente il flusso di dati, ottimizzare l'esecuzione delle operazioni e calcolare automaticamente i gradienti per l'addestramento. Internamente, i grafi computazionali rappresentano astrazioni per layer comuni come quelli convoluzionali, di pooling, ricorrenti e densi, con dati che includono attivazioni, pesi e bias rappresentati in tensori. Questa rappresentazione consente un calcolo efficiente, sfruttando la struttura del grafico per parallelizzare le operazioni e applicare ottimizzazioni.

Alcuni livelli comuni che i grafi computazionali potrebbero implementare includono layer convoluzionali, di attenzione, ricorrenti e densi. I layer fungono da astrazioni di livello superiore che definiscono calcoli specifici in cima alle operazioni di base rappresentate nel grafo. Ad esempio, un layer Denso esegue la moltiplicazione e l'addizione di matrici tra tensori di input, peso e bias. È importante notare che un layer opera su tensori come input e output; il layer stesso non è un tensore. Alcune differenze chiave tra layer e tensori sono:

- I layer contengono stati come pesi e bias. I tensori sono senza stato, contengono solo dati.
- I layer possono modificare lo stato interno durante l'addestramento. I tensori sono immutabili/di sola lettura.

- I layer sono astrazioni di livello superiore. I tensori sono a un livello inferiore e rappresentano direttamente dati e operazioni matematiche.
- I layer definiscono pattern di calcolo fissi. I tensori scorrono tra i livelli durante l'esecuzione.
- I layer vengono utilizzati indirettamente durante la creazione di modelli. I tensori scorrono tra i livelli durante l'esecuzione.

Quindi, mentre i tensori sono una struttura dati fondamentale che i layer consumano e producono, i layer hanno funzionalità aggiuntive per definire operazioni parametrizzate e addestramento. Mentre un layer configura le operazioni tensoriali in background, il layer rimane distinto dagli oggetti tensoriali. L'astrazione del layer rende la creazione e l'addestramento di reti neurali molto più intuitive. Questa astrazione consente agli sviluppatori di creare modelli impilando insieme questi layer senza implementare la logica del layer. Ad esempio, la chiamata di `tf.keras.layers.Conv2D` in TensorFlow crea un layer convoluzionale. Il framework gestisce il calcolo delle convoluzioni, la gestione dei parametri, ecc. Ciò semplifica lo sviluppo del modello, consentendo agli sviluppatori di concentrarsi sull'architettura anziché sulle implementazioni di basso livello. Le astrazioni dei layer utilizzano implementazioni altamente ottimizzate per le prestazioni. Consentono inoltre la portabilità, poiché la stessa architettura può essere eseguita su backend hardware diversi come GPU e TPU.

Inoltre, i grafi computazionali includono funzioni di attivazione come ReLU, sigmoide e tanh che sono essenziali per le reti neurali e molti framework le forniscono come astrazioni standard. Queste funzioni introducono non linearità che consentono ai modelli di approssimare funzioni complesse. I framework le forniscono come operazioni semplici e predefinite che possono essere utilizzate durante la costruzione di modelli, ad esempio if.nn.relu in TensorFlow. Questa astrazione consente flessibilità, poiché gli sviluppatori possono facilmente scambiare le funzioni di attivazione per ottimizzare le prestazioni. Le attivazioni predefinite sono inoltre ottimizzate dal framework per un'esecuzione più rapida.

Negli ultimi anni, modelli come ResNets e MobileNets sono emersi come architetture popolari, con i framework attuali che li pre-confezionano come grafi computazionali. Invece di preoccuparsi dei dettagli, gli sviluppatori possono utilizzarli come punto di partenza, personalizzandoli secondo necessità sostituendo i layer. Ciò semplifica e velocizza lo sviluppo del modello, evitando di reinventare le architetture da zero. I modelli predefiniti includono implementazioni ben collaudate e ottimizzate che garantiscono buone prestazioni. Il loro design modulare consente inoltre di trasferire le funzionalità apprese a nuove attività tramite apprendimento tramite trasferimento. Queste architetture predefinite forniscono i mattoni ad alte prestazioni per creare rapidamente modelli robusti.

Queste astrazioni di layer, funzioni di attivazione e architetture predefinite fornite dai framework costituiscono un grafo computazionale. Quando un utente definisce un layer in un framework (ad esempio, `tf.keras.layers.Dense()`), il framework configura nodi e bordi del grafo computazionale per rappresentare tale layer. I parametri del layer come pesi e bias diventano variabili nel grafo. I calcoli del layer diventano nodi operativi (come x e y nella figura sopra). Quando si chiama una funzione di attivazione come `tf.nn.relu()`, il framework aggiunge un nodo operativo ReLU al grafo. Le architetture predefinite sono solo sottografi preconfigurati che possono essere inseriti nel grafo del modello. Quindi, la definizione del modello tramite astrazioni di alto livello crea un grafo computazionale: i livelli, le attivazioni e le architetture che utilizziamo diventano nodi e rami del grafo.

Costruiamo implicitamente un grafo computazionale quando definiamo un'architettura di rete neurale in un framework. Il framework utilizza questo grafo per determinare le operazioni da eseguire durante l'addestramento e l'inferenza. I grafi computazionali offrono diversi

vantaggi rispetto al codice grezzo e questa è una delle funzionalità principali offerte da un buon framework di ML:

- Rappresentazione esplicita del flusso di dati e delle operazioni
- Capacità di ottimizzare il grafo prima dell'esecuzione
- Differenziazione automatica per il training
- Agnosticismo linguistico: il grafo può essere tradotto per essere eseguito su GPU, TPU, ecc.
- Portabilità: il grafo può essere serializzato, salvato e ripristinato in seguito

I grafi computazionali sono i componenti fondamentali dei framework di ML. La definizione del modello tramite astrazioni di alto livello crea un grafo computazionale: i layer, le attivazioni e le architetture che utilizziamo diventano nodi e rami del grafico. I compilatori e gli ottimizzatori del framework operano su questo grafo per generare codice eseguibile. Le astrazioni forniscono un'API intuitiva per gli sviluppatori per la creazione di grafi computazionali. Sotto, ci sono ancora grafi! Quindi, anche se non si possono manipolare direttamente i grafi come utente del framework, consentono di eseguire ad alto livello e in modo efficiente le specifiche del modello. Le astrazioni semplificano la creazione del modello, mentre i grafi computazionali la rendono possibile.

6.4.2.2 Grafi Statici vs. Dinamici

I framework di deep learning hanno tradizionalmente seguito uno dei due approcci per esprimere grafi computazionali.

Grafi statici (declare-then-execute): Con questo modello, l'intero grafo computazionale deve essere definito in anticipo prima di eseguirlo. Tutte le operazioni e le dipendenze dei dati devono essere specificate durante la fase di dichiarazione. TensorFlow originariamente seguiva questo approccio statico: i modelli venivano definiti in un contesto separato e poi veniva creata una sessione per eseguirli. Il vantaggio dei grafi statici è che consentono un'ottimizzazione più aggressiva poiché il framework può vedere il grafo completo. Tuttavia, tende anche a essere meno flessibile per la ricerca e l'interattività. Le modifiche al grafo richiedono la nuova dichiarazione del modello completo.
Per esempio:

```
x = tf.placeholder(tf.float32)
y = tf.matmul(x, weights) + biases
```

In questo esempio, x è un segnaposto per i dati di input e y è il risultato di un'operazione di moltiplicazione di matrici seguita da un'addizione. Il modello è definito in questa fase di dichiarazione, in cui tutte le operazioni e le variabili devono essere specificate in anticipo. Una volta definito l'intero grafo, il framework lo compila e lo ottimizza. Ciò significa che i passaggi computazionali sono definitivamente "scolpiti" e il framework può applicare varie ottimizzazioni per migliorare l'efficienza e le prestazioni. Quando in seguito si esegue il grafo, si forniscono i tensori di input effettivi e le operazioni predefinite vengono eseguite nella sequenza ottimizzata.
Questo approccio è simile alla creazione di un progetto in cui ogni dettaglio è pianificato prima dell'inizio della costruzione. Sebbene ciò consenta potenti ottimizzazioni, significa anche che qualsiasi modifica al modello richiede la ridefinizione dell'intero grafo da zero.

Grafi dinamici (define-by-run): A differenza della dichiarazione (di tutto) prima e dell'esecuzione poi, il grafo viene creato dinamicamente durante l'esecuzione. Non esiste una fase di dichiarazione separata: le operazioni vengono eseguite immediatamente come definite. Questo stile è imperativo e flessibile, facilitando la sperimentazione.
PyTorch utilizza grafi dinamici, creandoli al volo mentre avviene l'esecuzione. Ad esempio, si consideri il seguente frammento di codice, in cui il grafo viene creato durante l'esecuzione:

```
x = torch.randn(4,784)
y = torch.matmul(x, weights) + biases
```

L'esempio sopra non ha fasi separate di compilazione/build/esecuzione. Le operazioni definiscono ed eseguono immediatamente. Con i grafi dinamici, la definizione è intrecciata con l'esecuzione, fornendo un flusso di lavoro più intuitivo e interattivo. Tuttavia, lo svantaggio è che c'è meno potenziale di ottimizzazione poiché il framework vede solo il grafo mentre viene creato. Figura 6.7 mostra le differenze tra un grafo di calcolo statico e uno dinamico.

Figura 6.7: Confronto tra grafi statici e dinamici. Fonte: Dev

Di recente, la distinzione si è offuscata poiché i framework adottano entrambe le modalità. TensorFlow 2.0 passa automaticamente alla modalità di grafo dinamico, consentendo agli utenti di lavorare con quelli statici quando necessario. La dichiarazione dinamica offre flessibilità e facilità d'uso, rendendo i framework più intuitivi, mentre i grafi statici forniscono vantaggi di ottimizzazione a scapito dell'interattività. Il framework ideale bilancia questi approcci. Tabella 6.2 confronta i pro e i contro dei grafi di esecuzione statici e dinamici:

Tabella 6.2: Confronto tra Grafi di Esecuzione Statici (Declare-then-execute) e Dinamici (Define-by-run), evidenziandone i rispettivi pro e contro.

Grafo di esecuzione	Pro	Contro
Statico (Declare-then-execute)	• Abilita le ottimizzazioni del grafo visualizzando il modello completo in anticipo • Può esportare e distribuire grafici congelati • Il grafo è impacchettato indipendentemente dal codice	• Meno flessibile per la ricerca e l'iterazione • Le modifiche richiedono la ricostruzione del grafo • L'esecuzione ha fasi di compilazione ed esecuzione separate
Dinamico (Define-by-run)	• Stile imperativo intuitivo come il codice Python • Alterna la creazione del grafo con l'esecuzione • Facile da modificare i grafi	• Più difficile da ottimizzare senza un grafo completo • Possibili rallentamenti dalla creazione del

• Il debug si adatta perfettamente al flusso di lavoro	grafo durante l'esecuzione • Può richiedere più memoria

6.4.3 Tool della Pipeline dei Dati

I grafi computazionali possono essere validi solo quanto i dati da cui apprendono e su cui lavorano. Pertanto, alimentare i dati di training in modo efficiente è fondamentale per ottimizzare le prestazioni della "deep neural network" [rete neurale profonda], sebbene spesso venga trascurata come una delle funzionalità principali. Molti framework di IA moderni forniscono pipeline specializzate per acquisire, elaborare e aumentare i set di dati per il training del modello.

6.4.3.1 Loader dei Dati

Al centro di queste pipeline ci sono i "data loader", che gestiscono esempi di training di lettura da fonti come file, database e storage di oggetti. I data loader facilitano il caricamento e la pre-elaborazione efficienti dei dati, cruciali per i modelli di deep learning. Ad esempio, la pipeline di caricamento dati tf.data di TensorFlow è progettata per gestire questo processo. A seconda dell'applicazione, i modelli di deep learning richiedono diversi formati di dati come file CSV o cartelle di immagini. Alcuni formati popolari includono:

- **CSV**: Un formato versatile e semplice spesso utilizzato per dati tabulari.
- **TFRecord**: Formato proprietario di TensorFlow, ottimizzato per le prestazioni.
- **Parquet**: Archiviazione a colonne, che offre compressione e recupero dati efficienti.
- **JPEG/PNG**: Comunemente utilizzato per dati immagine.
- **WAV/MP3**: Formati prevalenti per dati audio.

Esempi di batch di data loader per sfruttare il supporto di vettorizzazione nell'hardware. Il "batching" si riferisce al raggruppamento di più dati per l'elaborazione simultanea, sfruttando le capacità di calcolo vettorizzate di hardware come le GPU. Sebbene le dimensioni tipiche dei batch siano comprese tra 32 e 512 esempi, la dimensione ottimale spesso dipende dall'ingombro di memoria dei dati e dai vincoli hardware specifici. I loader avanzati possono trasmettere in streaming set di dati virtualmente illimitati da dischi e archivi cloud. Trasmettono in streaming grandi dataset da dischi o reti anziché caricarli completamente in memoria, consentendo dimensioni illimitate.

I data loader possono anche mescolare i dati tra "epoche" per la randomizzazione e le funzionalità di preelaborazione in parallelo con l'addestramento del modello per accelerarne il processo. Mescolare casualmente l'ordine degli esempi tra epoche di training riduce il bias e migliora la generalizzazione.

I data loader supportano anche strategie di "caching" e "prefetching" per ottimizzare la distribuzione dei dati per un addestramento del modello rapido e fluido. Il caching [memorizzazione nella cache] dei batch preelaborati consente di riutilizzarli in modo efficiente durante più fasi di addestramento ed elimina l'elaborazione ridondante. Il prefetching, al contrario, comporta il precaricamento dei batch successivi, assicurando che il modello non resti mai inattivo in attesa di dati.

6.4.4 Data Augmentation

Framework di apprendimento automatico come TensorFlow e PyTorch forniscono strumenti per semplificare e snellire il processo di "data augmentation" [aumento dei dati], migliorando l'efficienza dell'espansione sintetica dei set di dati. Questi framework offrono funzionalità integrate per applicare trasformazioni casuali, come capovolgimento, ritaglio, rotazione, modifica del colore e aggiunta di rumore per le immagini. Per i dati audio, gli aumenti comuni

comportano la miscelazione di clip con rumore di fondo o la modulazione di velocità, tono e volume.
Integrando gli strumenti di "augmentation" nella pipeline di dati, i framework consentono di applicare queste trasformazioni al volo durante ogni epoca di addestramento. Questo approccio incrementa la variazione nella distribuzione dei dati di addestramento, riducendo così l'overfitting e migliorando la generalizzazione del modello. Figura 6.8 mostra i casi di overfitting e underfitting. L'uso di "data loader" performanti in combinazione con ampie capacità di "augmentation" consente ai professionisti di alimentare in modo efficiente set di dati massicci e vari alle reti neurali.

Figura 6.8: Overfitting e underfitting. Fonte: Aquarium Learning

Queste pipeline di dati "hands-off" rappresentano un miglioramento significativo in termini di usabilità e produttività. Consentono agli sviluppatori di concentrarsi maggiormente sull'architettura del modello e meno sulla manipolazione dei dati durante l'addestramento di modelli di deep learning.

6.4.5 Funzioni Loss e Algoritmi di Ottimizzazione

L'addestramento di una rete neurale è fondamentalmente un processo iterativo che cerca di minimizzare una funzione di loss [perdita]. L'obiettivo è di mettere a punto i pesi e i parametri del modello per produrre previsioni vicine alle vere etichette target. I framework di apprendimento automatico hanno notevolmente semplificato questo processo offrendo funzioni di loss [perdita] e algoritmi di ottimizzazione.
I framework di apprendimento automatico forniscono funzioni di perdita implementate che sono necessarie per quantificare la differenza tra le previsioni del modello e i valori reali. Diversi set di dati richiedono una diversa funzione di perdita per funzionare correttamente, poiché tale funzione indica al computer l'"obiettivo" a cui mirare. Le funzioni di perdita comunemente utilizzate includono il "Mean Squared Error (MSE)" [errore quadratico medio] per le attività di regressione, la "Cross-Entropy Loss" per le attività di classificazione, e la Kullback-Leibler (KL) per i modelli probabilistici. Ad esempio, tf.keras.losses di TensorFlow contiene una serie di queste funzioni di perdita comunemente utilizzate.
Gli algoritmi di ottimizzazione vengono utilizzati per trovare in modo efficiente il set di parametri del modello che minimizzano la funzione di perdita, assicurando che il modello funzioni bene sui dati di training e si generalizzi a nuovi dati. I framework moderni sono dotati di implementazioni efficienti di diversi algoritmi di ottimizzazione, molti dei quali sono varianti della "discesa del gradiente" con metodi stocastici e tassi di apprendimento adattivo. Alcuni

esempi di queste varianti sono Stochastic Gradient Descent, Adagrad, Adadelta e Adam. L'implementazione di tali varianti è fornita in tf.keras.optimizers. Ulteriori informazioni con esempi chiari sono disponibili nella sezione Training dell'IA.

6.4.6 Supporto al Training del Modello

È richiesta una fase di compilazione prima di addestrare un modello di rete neurale definito. Durante questa fase, l'architettura di alto livello della rete neurale viene trasformata in un formato eseguibile ottimizzato. Questo processo comprende diverse fasi. La prima fase consiste nel costruire il grafo computazionale, che rappresenta tutte le operazioni matematiche e il flusso di dati all'interno del modello. Ne abbiamo discusso in precedenza. Durante l'addestramento, l'attenzione è rivolta all'esecuzione del grafo computazionale. A ogni parametro all'interno del grafo, come pesi e bias, viene assegnato un valore iniziale. A seconda del metodo di inizializzazione scelto, questo valore potrebbe essere casuale o basato su una logica predefinita.

Il passaggio critico successivo è l'allocazione della memoria. La memoria essenziale è riservata alle operazioni del modello sia su CPU che su GPU, garantendo un'elaborazione efficiente dei dati. Le operazioni del modello vengono poi mappate sulle risorse hardware disponibili, in particolare GPU o TPU, per accelerare l'elaborazione. Una volta completata la compilazione, il modello viene preparato per l'addestramento.

Il processo di addestramento impiega vari strumenti per migliorare l'efficienza. L'elaborazione batch è comunemente utilizzata per massimizzare la produttività computazionale. Tecniche come la vettorizzazione consentono operazioni su interi array di dati anziché procedere elemento per elemento, il che aumenta la velocità. Ottimizzazioni come la "kernel fusion" (fare riferimento al capitolo Ottimizzazioni) amalgamano più operazioni in un'unica azione, riducendo al minimo il sovraccarico computazionale. Le operazioni possono anche essere segmentate in fasi, facilitando l'elaborazione simultanea di diversi mini-batch in varie parti.

I framework eseguono costantemente il checkpoint dello stato, preservando le versioni intermedie del modello durante l'addestramento. Ciò garantisce che i progressi vengano recuperati in caso di interruzione e che l'addestramento possa essere ripreso dall'ultimo checkpoint. Inoltre, il sistema monitora attentamente le prestazioni del modello rispetto a un set di dati di convalida. Se il modello inizia a sovradimensionarsi (se le sue prestazioni sul set di convalida diminuiscono), l'addestramento viene automaticamente interrotto, conservando risorse computazionali e tempo.

I framework di ML incorporano una combinazione di compilazione del modello, metodi di elaborazione batch avanzati e utilità come il checkpoint e l'arresto anticipato. Queste risorse gestiscono gli aspetti complessi delle prestazioni, consentendo ai professionisti di concentrarsi sullo sviluppo e l'addestramento del modello. Di conseguenza, gli sviluppatori sperimentano sia velocità che facilità quando utilizzano le capacità delle reti neurali.

6.4.7 Validazione e Analisi

Dopo aver addestrato i modelli di deep learning, i framework forniscono utilità per valutare le prestazioni e ottenere informazioni sul funzionamento dei modelli. Questi strumenti consentono una sperimentazione e un debug disciplinati.

6.4.7.1 Metriche di Valutazione

I framework includono implementazioni di comuni metriche di valutazione per la convalida:

- Accuratezza: Frazione di previsioni corrette complessive. Sono ampiamente utilizzate per la classificazione.
- Precisione: Delle previsioni positive, quante erano positive. Utile per set di dati sbilanciati.

- Richiamo: Dei positivi effettivi, quanti ne abbiamo previsti correttamente? Misura della Completezza.
- Punteggio F1: Media armonica di precisione e richiamo. Combina entrambe le metriche.
- AUC-ROC - Area sotto la curva ROC. Sono utilizzate per l'analisi della soglia di classificazione.
- MAP - Mean Average Precision. Valuta le previsioni classificate nel recupero/rilevamento.
- Matrice di Confusione: Matrice che mostra i veri positivi, i veri negativi, i falsi positivi e i falsi negativi. Fornisce una visione più dettagliata delle prestazioni di classificazione.

Queste metriche quantificano le prestazioni del modello sui dati di convalida per il confronto.

6.4.7.2 Visualizzazione

Gli strumenti di visualizzazione forniscono informazioni sui modelli:

- Curve di perdita: Tracciano la perdita di training e validazione nel tempo per individuare l'overfitting.
- Loss curves [Griglie di attivazione]: Illustrano le funzionalità apprese dai filtri convoluzionali.
- Projection [Proiezione]: Riduce la dimensionalità per una visualizzazione intuitiva.
- Precision-recall curves [Curve di richiamo della precisione]: Valutano i compromessi di classificazione. Figura 6.9 mostra un esempio di una curva di precisione-richiamo.

Figura 6.9: Lettura di una curva "precision-recall". Fonte: AIM

Strumenti come TensorBoard per TensorFlow e TensorWatch per PyTorch consentono metriche e visualizzazioni in tempo reale durante il training.

6.4.8 Programmazione differenziabile

I metodi di addestramento per il machine learning come la backpropagation si basano sulla modifica della funzione di perdita rispetto alla modifica dei pesi (che essenzialmente è la definizione di derivata). Pertanto, la capacità di addestrare rapidamente ed efficientemente grandi modelli di machine learning si basa sulla capacità del computer di prendere derivate. Ciò rende la programmazione differenziabile uno degli elementi più importanti di un framework di apprendimento automatico.

Possiamo utilizzare quattro metodi principali per far sì che i computer prendano derivate. Innanzitutto, possiamo calcolare manualmente le derivate a mano e inserirle nel computer. Questo diventerebbe rapidamente un incubo con molti layer di reti neurali se dovessimo calcolare manualmente tutte le derivate nei passaggi di backpropagation. Un altro metodo è la differenziazione simbolica utilizzando sistemi di computer algebrici come Mathematica, che può introdurre un layer di inefficienza, poiché è necessario un livello di astrazione per prendere le derivate. Le derivate numeriche, la pratica di approssimare i gradienti utilizzando metodi di differenze finite, soffrono di molti problemi, tra cui elevati costi computazionali e dimensioni della griglia più grandi, che portano a molti errori. Ciò porta alla differenziazione automatica, che sfrutta le funzioni primitive che i computer utilizzano per rappresentare le operazioni per ottenere una derivata esatta. Con la differenziazione automatica, la complessità computazionale del calcolo del gradiente è proporzionale al calcolo della funzione stessa. Le complessità della differenziazione automatica non sono gestite dagli utenti finali al momento, ma le risorse per saperne di più possono essere trovate ampiamente, ad esempio qui. La differenziazione automatica e la programmazione differenziabile di oggi sono onnipresenti e vengono eseguite in modo efficiente e automatico dai moderni framework di machine learning.

6.4.9 Accelerazione Hardware

La tendenza a formare e distribuire continuamente modelli di apprendimento automatico più grandi ha reso necessario il supporto dell'accelerazione hardware per le piattaforme di machine-learning. Figura 6.10 mostra il gran numero di aziende che offrono acceleratori hardware in diversi domini, come il machine learning "Very Low Power" e quello "Embedded". I "deep layer" delle reti neurali richiedono molte moltiplicazioni di matrici, che attraggono hardware in grado di calcolare rapidamente e in parallelo tali operazioni. In questo panorama, due architetture hardware, GPU e TPU, sono emerse come scelte principali per l'addestramento di modelli di apprendimento automatico.

L'uso di acceleratori hardware è iniziato con AlexNet, che ha aperto la strada a lavori futuri per utilizzare le GPU come acceleratori hardware per l'addestramento di modelli di visione artificiale. Le GPU, o "Graphics Processing Units" [unità di elaborazione grafica], eccellono nella gestione di molti calcoli contemporaneamente, il che le rende ideali per le operazioni matriciali centrali per l'addestramento delle reti neurali. La loro architettura, progettata per il rendering della grafica, è perfetta per le operazioni matematiche richieste nell'apprendimento automatico. Sebbene siano molto utili per le attività di apprendimento automatico e siano state implementate in molte piattaforme hardware, le GPU sono comunque di uso generale in quanto possono essere utilizzate per altre applicazioni.

D'altro canto, le Tensor Processing Units (TPU) sono unità hardware progettate specificamente per le reti neurali. Si concentrano sull'operazione di "moltiplicazione e accumulazione" (MAC) e il loro hardware è costituito da una grande matrice hardware che contiene elementi che calcolano in modo efficiente l'operazione MAC. Questo concetto, chiamato systolic array architecture, è stato ideato da Kung e Leiserson (1979), ma ha dimostrato di essere una struttura utile per calcolare in modo efficiente i prodotti matriciali e altre operazioni all'interno delle reti neurali (come le convoluzioni).

Sebbene le TPU possano ridurre drasticamente i tempi di addestramento, presentano anche degli svantaggi. Ad esempio, molte operazioni all'interno dei framework di apprendimento automatico (principalmente TensorFlow in questo caso, poiché la TPU si integra direttamente con esso) non sono supportate dalle TPU. Non possono inoltre supportare operazioni personalizzate dai framework di apprendimento automatico e la progettazione della rete deve essere strettamente allineata alle capacità hardware.

Oggi, le GPU NVIDIA dominano il training, supportate da librerie software come CUDA, cuDNN e TensorRT. I framework includono anche ottimizzazioni per massimizzare le prestazioni su questi tipi di hardware, come l'eliminazione di connessioni non importanti e la fusione di layer. La combinazione di queste tecniche con l'accelerazione hardware fornisce una maggiore efficienza. Per l'inferenza, l'hardware si sta spostando sempre di più verso ASIC e SoC ottimizzati. Le TPU di Google accelerano i modelli nei data center, mentre Apple, Qualcomm, la famiglia NVIDIA Jetson e altri ora producono chip "mobili" incentrati sull'intelligenza artificiale.

Figura 6.10: Aziende che offrono acceleratori hardware di ML. Fonte: Gradient Flow.

6.5 Funzionalità Avanzate

Oltre a fornire gli strumenti essenziali per il training di modelli di apprendimento automatico, i framework offrono anche funzionalità avanzate. Queste funzionalità includono la distribuzione del training su diverse piattaforme hardware, la facile messa a punto di grandi modelli pre-addestrati e l'esemplificazione del "federated learning". L'implementazione di queste funzionalità in modo indipendente sarebbe altamente complessa e richiederebbe molte risorse, ma i framework semplificano questi processi, rendendo le tecniche avanzate di apprendimento automatico più accessibili.

6.5.1 Training distribuito

Poiché i modelli di apprendimento automatico sono diventati più grandi nel corso degli anni, è diventato essenziale per i modelli di grandi dimensioni utilizzare più nodi di elaborazione nel processo di training. Questo processo, l'apprendimento distribuito, ha consentito maggiori capacità di training, ma ha anche imposto problemi nell'implementazione.
Possiamo considerare tre diversi modi per distribuire il lavoro di training dei modelli di apprendimento automatico su più nodi di elaborazione. Il partizionamento dei dati di input (o parallelismo dei dati) si riferisce a più processori che eseguono lo stesso modello su diverse partizioni di input. Questa è l'implementazione più semplice ed è disponibile per molti framework di machine learning. La distribuzione più impegnativa del lavoro è rappresentata dal parallelismo del modello, che si riferisce a più nodi di elaborazione che lavorano su parti diverse del modello, e dal parallelismo del modello pipelined, che si riferisce a più nodi di

elaborazione che lavorano su diversi layer del modello sullo stesso input. Gli ultimi due menzionati qui sono aree di ricerca attive.
I framework di ML che supportano l'apprendimento distribuito includono TensorFlow (tramite il suo modulo tf.distribute), PyTorch (tramite i suoi moduli torch.nn.DataParallel e torch.nn.DistributedDataParallel) e MXNet (tramite la sua API gluon).

6.5.2 Conversione del Modello

I modelli di machine learning hanno vari metodi per essere rappresentati e utilizzati in diversi framework e per diversi tipi di dispositivi. Ad esempio, un modello può essere convertito per essere compatibile con i framework di inferenza all'interno del dispositivo mobile. Il formato di default per i modelli TensorFlow sono i file di checkpoint contenenti pesi e architetture, necessari per riaddestrare i modelli. Tuttavia, i modelli vengono in genere convertiti nel formato TensorFlow Lite per la distribuzione mobile. TensorFlow Lite utilizza una rappresentazione compatta del "flat buffer" e ottimizzazioni per un'inferenza rapida su hardware mobile, eliminando tutto il bagaglio non necessario associato ai metadati di addestramento, come le strutture dei file di checkpoint.
Le ottimizzazioni del modello come la quantizzazione (vedere il capitolo Ottimizzazioni) possono ottimizzare ulteriormente i modelli per architetture target come i dispositivi mobili. Ciò riduce la precisione di pesi e attivazioni a `uint8` o a `int8` per un ingombro ridotto e un'esecuzione più rapida con acceleratori hardware supportati. Per la quantizzazione post-training, il convertitore di TensorFlow gestisce automaticamente analisi e conversione.
Framework come TensorFlow semplificano la distribuzione di modelli addestrati su dispositivi IoT mobili ed embedded tramite API di conversione semplici per il formato TFLite e la quantizzazione. La conversione pronta all'uso consente un'inferenza ad alte prestazioni su dispositivi mobili senza l'onere dell'ottimizzazione manuale. Oltre a TFLite, altri target comuni includono TensorFlow.js per la distribuzione Web, TensorFlow Serving per i servizi cloud e TensorFlow Hub per l'apprendimento tramite trasferimento. Le utility di conversione di TensorFlow gestiscono questi scenari per semplificare i flussi di lavoro end-to-end.
Ulteriori informazioni sulla conversione dei modelli in TensorFlow sono linkate qui.

6.5.3 AutoML, No-Code/Low-Code ML

In molti casi, l'apprendimento automatico può avere una barriera d'ingresso relativamente alta rispetto ad altri campi. Per addestrare e distribuire con successo modelli, è necessario avere una comprensione critica di una varietà di discipline, dalla scienza dei dati (elaborazione dei dati, pulizia dei dati), strutture di modelli (ottimizzazione degli iperparametri, architettura delle reti neurali), hardware (accelerazione, elaborazione parallela) e altro a seconda del problema in questione. La complessità di questi problemi ha portato all'introduzione di framework come AutoML, che cerca di rendere "l'apprendimento automatico disponibile anche a chi non è esperto di apprendimento automatico" e di "automatizzare la ricerca nell'apprendimento automatico". Hanno creato AutoWEKA, che aiuta nel complesso processo di selezione degli iperparametri, e Auto-sklearn e Auto-pytorch, un'estensione di AutoWEKA nelle popolari librerie sklearn e PyTorch.
Mentre questi sforzi per automatizzare parti delle attività di apprendimento automatico sono in corso, altri si sono concentrati sulla semplificazione dei modelli tramite l'implementazione di apprendimento automatico "no-code" [senza codice]/low-code [a basso codice], utilizzando un'interfaccia drag-and-drop con un'interfaccia utente di facile navigazione. Aziende come Apple, Google e Amazon hanno già creato queste piattaforme di facile utilizzo per consentire agli utenti di costruire modelli di apprendimento automatico che possono essere integrati nel loro ecosistema.

Questi passaggi per rimuovere le barriere all'ingresso continuano a democratizzare il machine learning, semplificano l'accesso per i principianti e semplificano il flusso di lavoro per gli esperti.

6.5.4 Metodi di Apprendimento Avanzati

6.5.4.1 Il Transfer Learning

Il "transfer learning" è la pratica di utilizzare le conoscenze acquisite da un modello pre-addestrato per addestrare e migliorare le prestazioni di un modello per un'attività diversa. Ad esempio, modelli come MobileNet e ResNet vengono addestrati sul set di dati ImageNet. Per fare ciò, si può congelare il modello pre-addestrato, utilizzandolo come un estrattore di feature per addestrare un modello molto più piccolo costruito sopra l'estrazione di feature. Si può anche mettere a punto l'intero modello per adattarlo al nuovo compito. I framework di apprendimento automatico semplificano il caricamento di modelli pre-addestrati, il congelamento di layer specifici e l'addestramento di layer personalizzati in cima. Semplificano questo processo fornendo API intuitive e un facile accesso a grandi repository di modelli pre-addestrati.

Il transfer learning, pur essendo potente, presenta delle sfide. Un problema significativo è la potenziale incapacità del modello modificato di svolgere le sue attività originali dopo il transfer learning. Per affrontare queste sfide, i ricercatori hanno proposto varie soluzioni. Ad esempio, Zhizhong Li e Hoiem (2018) ha introdotto il concetto di "Imparare senza Dimenticare." nel suo articolo "Learning without Forgetting", che da allora è stato implementato nelle moderne piattaforme di machine learning. Figura 6.11 fornisce un'illustrazione semplificata del concetto di "transfer learning" [apprendimento per trasferimento]:

Figura 6.11: Trasferimento dell'apprendimento. Fonte: Tech Target

Come mostrato in Figura 6.11, l'apprendimento per trasferimento implica l'assunzione di un modello addestrato su un'attività (l'attività di origine) e l'adattamento per eseguire un'attività nuova e correlata (l'attività di target). Questo processo consente al modello di sfruttare le conoscenze acquisite dall'attività di origine, migliorando potenzialmente le prestazioni e

riducendo i tempi di addestramento sull'attività di destinazione. Tuttavia, come accennato in precedenza, è necessario prestare attenzione per garantire che il modello non "dimentichi" la sua capacità di eseguire l'attività originale durante questo processo.

6.5.4.2 Il Federated Learning

Il "Federated learning" di McMahan et al. (2017a) è una forma di elaborazione distribuita che prevede l'addestramento di modelli su dispositivi personali anziché la centralizzazione dei dati su un singolo server (Figura 12.7). Inizialmente, un modello globale di base viene addestrato su un server centrale per essere distribuito a tutti i dispositivi. Utilizzando questo modello di base, i dispositivi calcolano individualmente i gradienti e li inviano all'hub centrale. Intuitivamente, questo trasferisce i parametri del modello anziché i dati stessi. L'apprendimento federato migliora la privacy mantenendo i dati sensibili sui dispositivi locali e condividendo gli aggiornamenti del modello solo con un server centrale. Questo metodo è particolarmente utile quando si gestiscono dati sensibili o quando un'infrastruttura su larga scala non è praticabile.

Figura 6.12: Un approccio con server centralizzato al "federated learning". Fonte: NVIDIA.

Tuttavia, il federated learning deve affrontare sfide come garantire l'accuratezza dei dati, gestire dati non-IID (independent and identically distributed) [indipendenti e distribuiti in modo identico], gestire la produzione di dati non bilanciata e superare il sovraccarico della comunicazione e l'eterogeneità dei dispositivi. Anche i problemi di privacy e sicurezza, come gli attacchi di inversione del gradiente, pongono sfide significative.
I framework di apprendimento automatico semplificano l'implementazione dell'apprendimento federato fornendo gli strumenti e le librerie necessarie. Ad esempio, TensorFlow Federated (TFF) offre un framework open source per supportare l'apprendimento federato. TFF consente agli sviluppatori di simulare e implementare algoritmi di apprendimento federato, offrendo un core federato per operazioni di basso livello e API di alto livello per attività federate comuni. Si integra perfettamente con TensorFlow, consentendo l'uso di modelli e ottimizzatori TensorFlow in un ambiente federato. TFF supporta tecniche di aggregazione sicure per migliorare la privacy e consente la personalizzazione degli algoritmi di apprendimento federato. Sfruttando questi strumenti, gli sviluppatori possono distribuire in modo efficiente il training, perfezionare i modelli pre-addestrati e gestire le complessità intrinseche dell'apprendimento federato.

Sono stati sviluppati anche altri programmi open source come Flower per semplificare l'implementazione dell'apprendimento federato con vari framework di machine learning.

6.6 Specializzazione del Framework

Finora abbiamo parlato in generale dei framework di ML. Tuttavia, in genere, i framework sono ottimizzati in base alle capacità computazionali e ai requisiti applicativi dell'ambiente target, che vanno dal cloud all'edge ai dispositivi minuscoli. La scelta del framework giusto è fondamentale in base all'ambiente target per la distribuzione. Questa sezione fornisce una panoramica dei principali tipi di framework di IA su misura per ambienti cloud, edge e TinyML per aiutare a comprendere le somiglianze e le differenze tra questi ecosistemi.

6.6.1 Cloud

I framework di IA basati su cloud presuppongono l'accesso a un'ampia potenza di calcolo, memoria e risorse di archiviazione nel cloud. In genere supportano sia il training che l'inferenza. I framework di IA basati su cloud sono adatti per applicazioni in cui i dati possono essere inviati al cloud per l'elaborazione, come servizi di IA basati su cloud, analisi di dati su larga scala e applicazioni Web. I framework di IA cloud più diffusi includono quelli che abbiamo menzionato in precedenza, come TensorFlow, PyTorch, MXNet, Keras, ecc. Questi framework utilizzano GPU, TPU, training distribuito e AutoML per fornire IA scalabile. Concetti come model serving, MLOps e AIOps sono correlati all'operatività dell'IA nel cloud. L'IA cloud alimenta servizi come Google Cloud AI e consente il "transfer learning" tramite modelli pre-addestrati.

6.6.2 Edge

I framework Edge AI sono pensati per distribuire modelli di IA su dispositivi IoT, smartphone e server edge. I framework Edge AI sono ottimizzati per dispositivi con risorse di calcolo moderate, bilanciando potenza e prestazioni. I framework Edge AI sono ideali per applicazioni che richiedono elaborazione in tempo reale o quasi reale, tra cui robotica, veicoli autonomi e dispositivi intelligenti. I principali framework Edge AI includono TensorFlow Lite, PyTorch Mobile, CoreML e altri. Impiegano ottimizzazioni come compressione del modello, quantizzazione ed architetture di reti neurali efficienti. Il supporto hardware include CPU, GPU, NPU e acceleratori come Edge TPU. Edge AI consente casi d'uso come visione mobile, riconoscimento vocale e rilevamento di anomalie in tempo reale.

6.6.3 Embedded

I framework TinyML sono specializzati per distribuire modelli AI su dispositivi con risorse estremamente limitate, in particolare microcontrollori e sensori all'interno dell'ecosistema IoT. I framework TinyML sono progettati per dispositivi con risorse limitate, enfatizzando memoria minima e consumo energetico. I framework TinyML sono specializzati per casi d'uso su dispositivi IoT con risorse limitate per applicazioni di manutenzione predittiva, riconoscimento dei gesti e monitoraggio ambientale. I principali framework TinyML includono TensorFlow Lite Micro, uTensor e ARM NN. Ottimizzano modelli complessi per adattarli a kilobyte di memoria tramite tecniche come l'addestramento consapevole della quantizzazione e la precisione ridotta. TinyML consente il rilevamento intelligente su dispositivi alimentati a batteria, consentendo l'apprendimento collaborativo tramite apprendimento federato. La scelta del framework implica il bilanciamento delle prestazioni del modello e dei vincoli computazionali della piattaforma target, che sia cloud, edge o TinyML. Tabella 6.3 confronta i principali framework di IA negli ambienti cloud, edge e TinyML:

Tabella 6.3: Confronto dei tipi di framework per Cloud AI, Edge AI e TinyML.

Tipo di framework	Esempi	Tecnologie chiave	Casi d'uso
Cloud AI	TensorFlow, PyTorch, MXNet, Keras	GPU, TPU, addestramento distribuito, AutoML, MLOps	Servizi cloud, app Web, analisi di big data
Edge AI	TensorFlow Lite, PyTorch Mobile, Core ML	Ottimizzazione del modello, compressione, quantizzazione, architetture NN efficienti	App mobili, sistemi autonomi, elaborazione in tempo reale
TinyML	TensorFlow Lite Micro, uTensor, ARM NN	Training consapevole della quantizzazione, precisione ridotta, ricerca di architettura neurale	Sensori IoT, dispositivi indossabili, manutenzione predittiva, riconoscimento dei gesti

Differenze principali:

- Cloud AI sfrutta un'enorme potenza di calcolo per modelli complessi utilizzando GPU/TPU e training distribuito.
- Edge AI ottimizza i modelli per l'esecuzione locale su dispositivi edge con risorse limitate.
- TinyML adatta i modelli a una memoria estremamente bassa e calcola ambienti come i microcontrollori.

6.7 Framework di IA Embedded

6.7.1 Vincoli di Risorse

I sistemi embedded affrontano gravi limitazioni di risorse che pongono sfide uniche quando si distribuiscono modelli di machine learning rispetto alle piattaforme di elaborazione tradizionali. Ad esempio, le unità microcontrollore (MCU) comunemente utilizzate nei dispositivi IoT hanno spesso:

- **RAM** varia da decine di kilobyte a pochi megabyte. Il popolare MCU ESP8266 ha circa 80 KB di RAM a disposizione degli sviluppatori. Ciò contrasta con 8 GB o più su laptop e desktop tipici odierni.
- **Memoria Flash** varia da centinaia di kilobyte a pochi megabyte. Il microcontrollore Arduino Uno fornisce solo 32 KB di archiviazione del codice. I computer standard odierni hanno un'archiviazione su disco nell'ordine dei terabyte.
- **Potenza di elaborazione** da pochi MHz a circa 200 MHz. L'ESP8266 funziona a 80 MHz. Questo è di diversi ordini di grandezza più lento delle CPU multi-core multi-GHz nei server e nei laptop di fascia alta.

Questi vincoli rigorosi spesso rendono impossibile l'addestramento di modelli di apprendimento automatico direttamente sui microcontrollori. La RAM limitata impedisce la gestione di grandi set di dati per il training. L'uso di energia per l'addestramento esaurirebbe rapidamente anche i dispositivi alimentati a batteria. Al contrario, i modelli vengono addestrati su sistemi ricchi di risorse e distribuiti su microcontrollori per un'inferenza ottimizzata. Ma anche l'inferenza pone delle sfide:

1. **Dimensioni del Modello:** I modelli di intelligenza artificiale sono troppo grandi per adattarsi a dispositivi IoT ed embedded. Ciò richiede tecniche di compressione del modello, come quantizzazione, potatura e "knowledge distillation" distillazione della conoscenza. Inoltre, come vedremo, molti dei framework utilizzati dagli sviluppatori di intelligenza artificiale hanno grandi quantità di overhead e librerie integrate che i sistemi embedded non possono supportare.

2. **Complessità delle Attività:** Con solo decine di KB o pochi MB di RAM, i dispositivi IoT e i sistemi embedded sono limitati nella complessità delle attività che possono gestire. Le attività che richiedono grandi set di dati o algoritmi sofisticati, ad esempio LLM, che verrebbero eseguiti senza problemi su piattaforme di elaborazione tradizionali potrebbero non essere fattibili su sistemi embedded senza compressione o altre tecniche di ottimizzazione a causa delle limitazioni di memoria.
3. **Archiviazione ed Elaborazione dei Dati:** I sistemi embedded spesso elaborano i dati in tempo reale e potrebbero archiviarne solo piccole quantità localmente. Al contrario, i sistemi di elaborazione tradizionali possono contenere ed elaborare grandi set di dati in memoria, consentendo un'analisi più rapida delle operazioni sui dati e aggiornamenti in tempo reale.
4. **Sicurezza e Privacy:** La poca memoria limita anche la complessità degli algoritmi e dei protocolli di sicurezza, la crittografia dei dati, le protezioni da reverse engineering e altro che può essere implementato sul dispositivo. Ciò potrebbe rendere alcuni dispositivi IoT più vulnerabili agli attacchi.

Di conseguenza, le ottimizzazioni software specializzate e i framework ML su misura per i microcontrollori devono funzionare entro questi stretti limiti delle risorse. Tecniche di ottimizzazione intelligenti come quantizzazione, potatura e distillazione della conoscenza comprimono i modelli per adattarli alla memoria limitata (vedere la sezione Ottimizzazioni). Gli insegnamenti tratti dalla ricerca di architettura neurale aiutano a guidare la progettazione dei modelli.

I miglioramenti hardware come gli acceleratori ML dedicati sui microcontrollori aiutano anche ad alleviare i vincoli. Ad esempio, Hexagon DSP di Qualcomm accelera i modelli TensorFlow Lite sui chip mobili Snapdragon. Google Edge TPU racchiude le prestazioni ML in un piccolo ASIC per dispositivi edge. ARM Ethos-U55 offre un'inferenza efficiente sui microcontrollori di classe Cortex-M. Questi chip ML personalizzati sbloccano funzionalità avanzate per applicazioni con risorse limitate.

A causa della potenza di elaborazione limitata, è quasi sempre impossibile addestrare modelli di intelligenza artificiale su IoT o sistemi embedded. Invece, i modelli vengono addestrati su potenti computer tradizionali (spesso con GPU) e poi distribuiti sul dispositivo embedded per l'inferenza. TinyML si occupa specificamente di questo, assicurando che i modelli siano sufficientemente leggeri per l'inferenza in tempo reale su questi dispositivi limitati.

6.7.2 Framework e Librerie

I framework di intelligenza artificiale embedded sono strumenti software e librerie progettati per abilitare funzionalità di intelligenza artificiale e ML su sistemi embedded. Questi framework sono essenziali per portare l'intelligenza artificiale su dispositivi IoT, robotica e altre piattaforme di edge computing e sono progettati per funzionare dove risorse di elaborazione, memoria e consumo energetico sono limitati.

6.7.3 Sfide

Sebbene i sistemi embedded rappresentino un'enorme opportunità per l'implementazione dell'apprendimento automatico per abilitare capacità intelligenti in edge, questi ambienti con risorse limitate pongono sfide significative. A differenza dei tipici ambienti cloud o desktop ricchi di risorse computazionali, i dispositivi embedded introducono gravi limitazioni in termini di memoria, potenza di elaborazione, efficienza energetica e hardware specializzato. Di conseguenza, le tecniche e i framework di apprendimento automatico esistenti progettati per cluster di server con risorse abbondanti non si traducono direttamente nei sistemi

embedded. Questa sezione svela alcune delle sfide e delle opportunità per i sistemi embedded e i framework ML.

6.7.3.1 Ecosistema Frammentato

La mancanza di un framework ML unificato ha portato a un ecosistema altamente frammentato. Gli ingegneri di aziende come STMicroelectronics, NXP Semiconductors e Renesas hanno dovuto sviluppare soluzioni personalizzate su misura per le loro specifiche architetture di microcontrollori e DSP. Questi framework ad hoc richiedevano un'ampia ottimizzazione manuale per ogni piattaforma hardware di basso livello. Ciò ha reso estremamente difficile il porting dei modelli, richiedendo la riqualificazione per nuove architetture Arm, RISC-V o proprietarie.

6.7.3.2 Esigenze Hardware Disparate

Senza un framework condiviso, non esisteva un modo standard per valutare le capacità dell'hardware. Fornitori come Intel, Qualcomm e NVIDIA crearono soluzioni integrate, combinando modelli e migliorando software e hardware. Ciò rese difficile discernere i motivi del guadagni di prestazioni, se fosse merito dei nuovi progetti di chip come i core x86 a basso consumo di Intel o le ottimizzazioni software. Era necessario un framework standard affinché i fornitori potessero valutare le capacità del loro hardware in modo equo e riproducibile.

6.7.3.3 Mancanza di Portabilità

Con strumenti standardizzati, adattare modelli addestrati in framework comuni come TensorFlow o PyTorch per funzionare in modo efficiente sui microcontrollori era più facile. Richiedeva una traduzione manuale dispendiosa, in termini di tempo, dei modelli per l'esecuzione su DSP specializzati di aziende come CEVA o core Arm M-series a basso consumo. Nessuno strumento immediato consentiva l'implementazione portatile su diverse architetture.

6.7.3.4 Infrastruttura Incompleta

L'infrastruttura per supportare i flussi di lavoro di sviluppo dei modelli chiave doveva essere migliorata. È necessario un maggiore supporto per le tecniche di compressione per adattare modelli di grandi dimensioni a budget di memoria limitati. Mancavano strumenti per la quantizzazione per ridurre la precisione per un'inferenza più rapida. Le API standardizzate per l'integrazione nelle applicazioni erano incomplete. Mancavano funzionalità essenziali come il debugging sul dispositivo, le metriche e la profilazione delle prestazioni. Queste lacune hanno aumentato i costi e la difficoltà dello sviluppo ML embedded.

6.7.3.5 Nessun Benchmark Standard

Senza benchmark unificati, non esisteva un modo standard per valutare e confrontare le capacità di diverse piattaforme hardware di fornitori come NVIDIA, Arm e Ambiq Micro. Le valutazioni esistenti si basavano su benchmark proprietari pensati per mostrare i punti di forza di specifici chip. Ciò rendeva impossibile misurare i miglioramenti hardware in modo oggettivo, imparziale e imparziale. Il capitolo Benchmarking dell'IA affronta questo argomento in modo più dettagliato.

6.7.3.6 Test Minimi del Mondo Reale

Gran parte dei benchmark si basava su dati sintetici. Testare rigorosamente i modelli su applicazioni embedded nel mondo reale era difficile senza set di dati e benchmark standardizzati, sollevando dubbi su come le dichiarazioni sulle prestazioni si sarebbero tradotte in un utilizzo nel mondo reale. Erano necessari test più approfonditi per convalidare i chip in casi di utilizzo reali.

La mancanza di framework e infrastrutture condivisi ha rallentato l'adozione di TinyML, ostacolandone l'integrazione nei prodotti embedded. I recenti framework standard hanno

iniziato ad affrontare questi problemi attraverso una migliore portabilità, profilazione delle prestazioni e supporto per il benchmarking. Tuttavia, è ancora necessaria un'innovazione continua per consentire un'implementazione fluida e conveniente dell'IA nei dispositivi edge.

6.7.3.7 Riepilogo

L'assenza di framework, benchmark e infrastrutture standardizzati per ML embedded ne ha tradizionalmente ostacolato l'adozione. Tuttavia, sono stati compiuti recenti progressi nello sviluppo di framework condivisi come TensorFlow Lite Micro e suite di benchmark come MLPerf Tiny che mirano ad accelerare la proliferazione di soluzioni TinyML. Tuttavia, superare la frammentazione e la difficoltà dell'implementazione embedded rimane un processo in corso.

6.8 Esempi

Il deployment distribuzione di machine learning su microcontrollori e altri dispositivi embedded richiede spesso librerie software e framework appositamente ottimizzati per funzionare entro vincoli rigorosi di memoria, elaborazione e potenza. Esistono diverse opzioni per eseguire l'inferenza su hardware con risorse limitate, ciascuna con il proprio approccio all'ottimizzazione dell'esecuzione del modello. Questa sezione esplorerà le caratteristiche chiave e i principi di progettazione alla base di TFLite Micro, TinyEngine e CMSIS-NN, fornendo informazioni su come ogni framework affronta il complesso problema dell'esecuzione di reti neurali molto accurata ma efficiente sui microcontrollori. Mostrerà inoltre diversi approcci per l'implementazione di framework TinyML efficienti.
Tabella 6.4 riassume le principali differenze e somiglianze tra questi tre framework di inferenza di apprendimento automatico specializzati per sistemi embedded e microcontrollori.

Tabella 6.4: Confronto dei framework: TensorFlow Lite Micro, TinyEngine e CMSIS-NN

Framework	TensorFlow Lite Micro	TinyEngine	CMSIS-NN
Approccio	Basato su interprete	Compilazione statica	Kernel di reti neurali ottimizzati
Focus sull'hardware	Dispositivi embedded generali	Microcontrollori	Processori ARM Cortex-M
Supporto aritmetico	Virgola mobile	Virgola mobile, virgola fissa	Virgola mobile, virgola fissa
Supporto del modello	Modelli di rete neurale generale	Modelli co-progettati con TinyNAS	Tipi di livelli di rete neurale comuni
Impronta del codice	Più grande grazie all'inclusione di interprete e operazioni	Piccola, include solo le operazioni necessarie per il modello	Nativamente leggera
Latenza	Più alta grazie a overhead di interpretazione	Molto bassa grazie al modello compilato	focalizzato sulla bassa latenza
Gestione della memoria	Gestita dinamicamente da interprete	Ottimizzazione a livello di modello	Strumenti per un'allocazione efficiente

Approccio di ottimizzazione	Alcune funzionalità di e generazione del codice	Kernel specializzati, fusione di operatori	Ottimizzazioni di assemblaggio specifiche dell'architettura
Principali vantaggi	Flessibilità, portabilità, facile aggiornamento dei modelli	Massimizza le prestazioni, ottimizza l'utilizzo della memoria	Accelerazione hardware, API standardizzata, portabilità

Ne comprenderemo ciascuno in modo più dettagliato nelle sezioni seguenti.

6.8.1 Interprete

TensorFlow Lite Micro (TFLM) è un framework di inferenza di apprendimento automatico progettato per dispositivi embedded con risorse limitate. Utilizza un interprete per caricare ed eseguire modelli di apprendimento automatico, il che fornisce flessibilità e facilità di aggiornamento dei modelli sul campo (David et al. 2021).

Gli interpreti tradizionali spesso hanno un overhead di branching [diramazione] significativo, che può ridurre le prestazioni. Tuttavia, l'interpretazione del modello di machine learning trae vantaggio dall'efficienza dei kernel di lunga durata, in cui ogni runtime del kernel è relativamente grande e aiuta a mitigare l'overhead dell'interprete.

Un'alternativa a un motore di inferenza basato su interprete è quella di generare codice nativo da un modello durante l'esportazione. Ciò può migliorare le prestazioni, ma sacrifica portabilità e flessibilità, poiché il codice generato deve essere ricompilato per ogni piattaforma target e deve essere sostituito completamente per modificare un modello.

TFLM bilancia la semplicità della compilazione del codice e la flessibilità di un approccio basato su interprete includendo alcune funzionalità di generazione del codice. Ad esempio, la libreria può essere costruita esclusivamente da file sorgenti, offrendo gran parte della semplicità della compilazione associata alla generazione di codice, pur mantenendo i vantaggi di un framework che esegue il modello interpretandolo.

Un approccio basato su interprete offre diversi vantaggi rispetto alla generazione di codice per l'inferenza di apprendimento automatico su dispositivi embedded:

- **Flessibilità:** I modelli possono essere aggiornati sul campo senza ricompilare l'intera applicazione.
- **Portabilità:** L'interprete può essere utilizzato per eseguire modelli su diverse piattaforme target senza dover effettuare il porting del codice.
- **Efficienza della Memoria:** L'interprete può condividere il codice su più modelli, riducendo l'utilizzo della memoria.
- **Facilità di sviluppo:** Gli interpreti sono più facili da sviluppare e gestire rispetto ai generatori di codice.

TensorFlow Lite Micro è un framework potente e flessibile per l'inferenza di apprendimento automatico su dispositivi embedded. Il suo approccio basato su interprete offre diversi vantaggi rispetto alla generazione di codice, tra cui flessibilità, portabilità, efficienza della memoria e facilità di sviluppo.

6.8.2 Basati su Compilatore

TinyEngine è un framework di inferenza ML progettato specificamente per microcontrollori con risorse limitate. Utilizza diverse ottimizzazioni per consentire l'esecuzione di reti neurali molto accurate entro i vincoli rigorosi di memoria, elaborazione e archiviazione sui microcontrollori (J. Lin et al. 2020).

Mentre framework di inferenza come TFLite Micro utilizzano interpreti per eseguire il grafo della rete neurale in modo dinamico in fase di esecuzione, ciò aggiunge un overhead

significativo per quanto riguarda l'utilizzo della memoria per archiviare metadati, latenza di interpretazione e mancanza di ottimizzazioni. Tuttavia, TFLite sostiene che l'overhead è piccolo. TinyEngine elimina questo overhead utilizzando un approccio di generazione del codice. Analizza il grafo di rete durante la compilazione e genera codice specializzato per eseguire solo quel modello. Questo codice viene compilato in modo nativo nel binario dell'applicazione, evitando i costi di interpretazione in fase di esecuzione.
I framework ML convenzionali pianificano la memoria per layer, cercando di ridurre al minimo l'utilizzo per ogni layer separatamente. TinyEngine esegue la pianificazione a livello di modello anziché analizzare l'utilizzo della memoria tra i layer. Assegna una dimensione di buffer comune in base alle esigenze massime di memoria di tutti i layer. Questo buffer viene quindi condiviso in modo efficiente tra i layer per aumentare il riutilizzo dei dati.
TinyEngine è inoltre specializzato nei kernel per ogni layer tramite tecniche come operatori di tiling, unrolling e fusing. Ad esempio, genererà kernel di calcolo unrolled [srotolato] con il numero di loop necessari per una convoluzione 3x3 o 5x5. Questi kernel specializzati estraggono le massime prestazioni dall'hardware del microcontrollore. Utilizza convoluzioni depthwise [in profondità] ottimizzate per ridurre al minimo le allocazioni di memoria calcolando l'output di ogni canale posizionato sui dati del canale di input. Questa tecnica sfrutta la natura separabile dei canali delle convoluzioni depthwise per ridurre le dimensioni di picco della memoria.
Come TFLite Micro, il binario TinyEngine compilato include solo le operazioni necessarie per un modello specifico anziché tutte le operazioni possibili. Ciò si traduce in un footprint binario molto piccolo, mantenendo basse le dimensioni del codice per i dispositivi con limiti di memoria.
Una differenza tra TFLite Micro e TinyEngine è che quest'ultimo è co-progettato con "TinyNAS", un metodo di ricerca di architettura per modelli di microcontrollori simile al NAS differenziale per microcontrollori. L'efficienza di TinyEngine consente di esplorare modelli più grandi e accurati tramite NAS. Fornisce inoltre feedback a TinyNAS su quali modelli possono rientrare nei vincoli hardware.
Attraverso varie tecniche personalizzate, come la compilazione statica, la pianificazione basata sul modello, kernel specializzati e la co-progettazione con NAS, TinyEngine consente un'inferenza di deep learning ad alta precisione entro i vincoli di risorse rigorosi dei microcontrollori.

6.8.3 Libreria

CMSIS-NN, acronimo di Cortex Microcontroller Software Interface Standard for Neural Networks, è una libreria software ideata da ARM. Offre un'interfaccia standardizzata per distribuire l'inferenza di reti neurali su microcontrollori e sistemi embedded, concentrandosi sull'ottimizzazione per i processori ARM Cortex-M (Lai, Suda, e Chandra 2018a).

Kernel di Reti Neurali: CMSIS-NN ha kernel altamente efficienti che gestiscono operazioni fondamentali di reti neurali come convoluzione, pooling, layer completamente connessi e funzioni di attivazione. Si rivolge a un'ampia gamma di modelli di reti neurali supportando l'aritmetica a virgola mobile e fissa. Quest'ultima è particolarmente utile per i dispositivi con risorse limitate in quanto riduce i requisiti di memoria e di calcolo (Quantization).

Accelerazione Hardware: CMSIS-NN sfrutta la potenza delle istruzioni SIMD (Single Instruction, Multiple Data) disponibili su molti processori Cortex-M. Ciò consente l'elaborazione parallela di più elementi di dati all'interno di una singola istruzione, aumentando così l'efficienza computazionale. Alcuni processori Cortex-M dispongono di estensioni di Digital Signal Processing (DSP) che CMSIS-NN può sfruttare per l'esecuzione accelerata della rete neurale. La libreria include anche ottimizzazioni a livello di assembly su

misura per specifiche architetture di microcontrollori per migliorare ulteriormente le prestazioni.

API standardizzata: CMSIS-NN offre un'API coerente e astratta che protegge gli sviluppatori dalle complessità dei dettagli hardware di basso livello. Ciò semplifica l'integrazione dei modelli di rete neurale nelle applicazioni. Può anche comprendere strumenti o utilità per convertire i formati di modelli di rete neurale più diffusi in un formato compatibile con CMSIS-NN.

Gestione della Memoria: CMSIS-NN fornisce funzioni per un'allocazione e una gestione efficienti della memoria, il che è fondamentale nei sistemi embedded in cui le risorse di memoria sono scarse. Garantisce un utilizzo ottimale della memoria durante l'inferenza e, in alcuni casi, consente operazioni in loco per ridurre il sovraccarico di memoria.

Portabilità: CMSIS-NN è progettato per la portabilità su vari processori Cortex-M. Questo consente agli sviluppatori di scrivere codice che possa funzionare su diversi microcontrollori senza modifiche significative.

Bassa Latenza: CMSIS-NN riduce al minimo la latenza di inferenza, rendendolo una scelta ideale per applicazioni in tempo reale in cui è fondamentale prendere decisioni rapide.

Efficienza Energetica: La libreria è progettata con un focus sull'efficienza energetica, rendendola adatta per dispositivi alimentati a batteria e con vincoli energetici.

6.9 Scelta del Framework Giusto

La scelta del framework di machine learning giusto per una determinata applicazione richiede un'attenta valutazione di modelli, hardware e considerazioni software. Figura 6.13 fornisce un confronto tra diversi framework TensorFlow, che discuteremo più in dettaglio:

Model	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Training	Yes	No	No
Inference	Yes *(but inefficient on edge)*	Yes *(and efficient)*	Yes *(and even **more** efficient)*
How Many Ops	~1400	~130	~50
Native Quantization Tooling + Support	No	Yes	Yes

Figura 6.13: Confronto tra Framework TensorFlow - Generale. Fonte: TensorFlow.

Analizzando questi tre aspetti, modelli, hardware e software, come illustrato in Figura 6.13, gli ingegneri ML possono selezionare il framework ottimale e personalizzarlo in base alle esigenze per applicazioni ML su dispositivo efficienti e performanti. L'obiettivo è bilanciare complessità del modello, limitazioni hardware e integrazione software per progettare una pipeline ML su misura per dispositivi embedded e edge. Mentre esaminiamo le differenze mostrate in Figura 6.13, acquisiremo informazioni su come scegliere il framework giusto e comprenderemo cosa causa le variazioni tra i framework.

6.9.1 Modello

Figura 6.13 illustra le principali differenze tra le varianti di TensorFlow, in particolare in termini di operazioni supportate (op) e funzionalità. TensorFlow supporta molte più operazioni rispetto a TensorFlow Lite e TensorFlow Lite Micro, poiché viene in genere utilizzato per la ricerca o l'implementazione cloud, che richiedono un numero elevato di operatori e una maggiore flessibilità.
La figura dimostra chiaramente questa differenza nel supporto op tra i framework. TensorFlow Lite supporta operazioni selezionate per il training sul dispositivo, mentre TensorFlow Micro no. Inoltre, la figura mostra che TensorFlow Lite supporta forme dinamiche e training consapevole della quantizzazione, funzionalità assenti in TensorFlow Micro. Al contrario, sia TensorFlow Lite che TensorFlow Micro offrono strumenti e supporto di quantizzazione nativi. Qui, la quantizzazione si riferisce alla trasformazione di un programma ML in una rappresentazione approssimata con operazioni di precisione inferiore disponibili, una funzionalità cruciale per dispositivi embedded e edge con risorse computazionali limitate.

6.9.2 Software

Come mostrato in Figura 6.14, TensorFlow Lite Micro non supporta il sistema operativo, mentre TensorFlow e TensorFlow Lite sì. Questa scelta di progettazione per TensorFlow Lite Micro aiuta a ridurre il sovraccarico di memoria, a rendere i tempi di avvio più rapidi e a consumare meno energia. Invece, TensorFlow Lite Micro può essere utilizzato insieme a sistemi operativi in tempo reale (RTOS) come FreeRTOS, Zephyr e Mbed OS.
La figura evidenzia anche un'importante funzionalità di gestione della memoria: TensorFlow Lite e TensorFlow Lite Micro supportano la mappatura della memoria del modello, consentendo l'accesso diretto ai modelli dall'archiviazione flash anziché caricarli nella RAM. Al contrario, TensorFlow non offre questa capacità.

Software	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Needs an OS	Yes	Yes	No
Memory Mapping of Models	No	Yes	Yes
Delegation to accelerators	Yes	Yes	No

Figura 6.14: Confronto tra Framework TensorFlow - Software. Fonte: TensorFlow.

Un'altra differenza fondamentale è la "accelerator delegation" [delega dell'acceleratore]. TensorFlow e TensorFlow Lite supportano questa funzionalità, consentendo loro di pianificare il codice su acceleratori diversi. Tuttavia, TensorFlow Lite Micro non offre la delega dell'acceleratore, poiché i sistemi embedded tendono ad avere una gamma limitata di acceleratori specializzati.
Queste differenze dimostrano come ogni variante di TensorFlow sia ottimizzata per il suo ambiente di distribuzione di target, dai potenti server cloud ai dispositivi embedded con risorse limitate.

6.9.3 Hardware

TensorFlow Lite e TensorFlow Lite Micro hanno dimensioni binarie di base e footprint di memoria significativamente più piccoli rispetto a TensorFlow (vedere Figura 6.15). Ad esempio, un tipico binario TensorFlow Lite Micro è inferiore a 200 KB, mentre TensorFlow è

molto più grande. Ciò è dovuto agli ambienti con risorse limitate dei sistemi embedded. TensorFlow supporta x86, TPU e GPU come NVIDIA, AMD e Intel.

Hardware	TensorFlow	TensorFlow Lite	TensorFlow Lite Micro
Base Binary Size	3MB+	100KB	~10 KB
Base Memory Footprint	~5MB	300KB	20KB
Optimized Architectures	X86, TPUs, GPUs	Arm Cortex A, x86	Arm Cortex M, DSPs, MCUs

Figura 6.15: Confronto tra Framework TensorFlow - Hardware. Fonte: TensorFlow.

TensorFlow Lite supporta i processori Arm Cortex-A e x86 comunemente utilizzati su telefoni cellulari e tablet. Quest'ultimo è privo di tutta la logica di training non necessaria per l'implementazione sul dispositivo. TensorFlow Lite Micro fornisce supporto per core Arm Cortex M focalizzati sui microcontrollori come M0, M3, M4 e M7, nonché DSP come Hexagon e SHARC e MCU come STM32, NXP Kinetis, Microchip AVR.

6.9.4 Altri Fattori

La selezione del framework di IA appropriato è essenziale per garantire che i sistemi embedded possano eseguire in modo efficiente i modelli di IA. Diversi fattori chiave oltre a modelli, hardware e software dovrebbero essere presi in considerazione quando si valutano i framework IA per i sistemi embedded.
Altri fattori chiave da considerare quando si sceglie un framework di apprendimento automatico sono prestazioni, scalabilità, facilità d'uso, integrazione con strumenti di ingegneria dei dati, integrazione con strumenti di ottimizzazione dei modelli e supporto della community. Gli sviluppatori possono prendere decisioni informate e massimizzare il potenziale delle iniziative di apprendimento automatico comprendendo questi diversi fattori.

6.9.4.1 Prestazioni

Le prestazioni sono fondamentali nei sistemi embedded in cui le risorse di calcolo sono limitate. Valutare la capacità del framework di ottimizzare l'inferenza del modello per l'hardware embedded. La quantizzazione del modello e il supporto dell'accelerazione hardware sono cruciali per ottenere un'inferenza efficiente.

6.9.4.2 Scalabilità

La scalabilità è essenziale quando si considera la potenziale crescita di un progetto di IA embedded. Il framework dovrebbe supportare l'implementazione di modelli su vari dispositivi embedded, dai microcontrollori ai processori più potenti. Dovrebbe inoltre gestire senza problemi sia le distribuzioni su piccola che su larga scala.

6.9.4.3 Integrazione con Strumenti di Data Engineering

Gli strumenti di ingegneria dei dati sono essenziali per la pre-elaborazione dei dati e la gestione della pipeline. Un framework di intelligenza artificiale ideale per sistemi embedded dovrebbe integrarsi perfettamente con questi strumenti, consentendo un'efficiente acquisizione dei dati, trasformazione e addestramento del modello.

6.9.4.4 Integrazione con Strumenti di Ottimizzazione del Modello

L'ottimizzazione del modello garantisce che i modelli di intelligenza artificiale siano adatti per la distribuzione embedded. Valutare se il framework si integra con strumenti di

ottimizzazione del modello come TensorFlow Lite Converter o ONNX Runtime per facilitare la quantizzazione del modello e la riduzione delle dimensioni.

6.9.4.5 Facilità d'Uso

La facilità d'uso di un framework di IA ha un impatto significativo sull'efficienza dello sviluppo. Un framework con un'interfaccia intuitiva e una documentazione chiara riduce la curva di apprendimento degli sviluppatori. Si dovrebbe considerare se il framework supporta API di alto livello, consentendo agli sviluppatori di concentrarsi sulla progettazione del modello piuttosto che sui dettagli di implementazione di basso livello. Questo fattore è incredibilmente importante per i sistemi embedded, che hanno meno funzionalità di quelle a cui gli sviluppatori tipici potrebbero essere abituati.

6.9.4.6 Supporto della Community

Il supporto della community gioca un altro fattore essenziale. I framework con community attive e coinvolte spesso hanno basi di codice ben mantenute, ricevono aggiornamenti regolari e forniscono forum preziosi per la risoluzione dei problemi. Di conseguenza, anche il supporto della community gioca un ruolo nella facilità d'uso perché garantisce che gli sviluppatori abbiano accesso a una vasta gamma di risorse, tra cui tutorial e progetti di esempio. Il supporto della community fornisce una certa garanzia che il framework continuerà a essere supportato per futuri aggiornamenti. Ci sono solo pochi framework che soddisfano le esigenze di TinyML. TensorFlow Lite Micro è il più popolare e ha il maggior supporto della comunità.

6.10 Tendenze Future nei Framework ML

6.10.1 Decomposizione

Attualmente, lo stack del sistema ML è costituito da quattro astrazioni come mostrato in Figura 6.16, vale a dire (1) grafi computazionali, (2) programmi tensoriali, (3) librerie e runtime e (4) primitive hardware.

Figura 6.16: Quattro astrazioni negli attuali stack dei sistemi ML. Fonte: TVM.

Ciò ha portato a confini verticali (ad esempio, tra i livelli di astrazione) e orizzontali (ad esempio, approcci basati sulla libreria rispetto a quelli basati sulla compilazione per il calcolo dei tensori), che ostacolano l'innovazione per il ML. Il lavoro futuro nei framework ML può guardare alla rottura di questi confini. A dicembre 2021 è stato proposto Apache TVM Unity, che mirava a facilitare le interazioni tra i diversi livelli di astrazione (nonché le persone dietro di essi, come scienziati ML, ingegneri ML e ingegneri hardware) e a co-ottimizzare le decisioni in tutti e quattro i livelli di astrazione.

6.10.2 Compilatori e Librerie ad Alte Prestazioni

Con l'ulteriore sviluppo dei framework ML, continueranno a emergere compilatori e librerie ad alte prestazioni. Alcuni esempi attuali includono TensorFlow XLA e CUTLASS di Nvidia, che accelerano le operazioni di algebra lineare nei grafi computazionali, e TensorRT di Nvidia, che accelera e ottimizza l'inferenza.

6.10.3 ML per Framework ML

Possiamo anche usare il ML per migliorare i framework di ML in futuro. Alcuni usi correnti di ML per framework ML includono:

- Ottimizzazione degli iperparametri tramite tecniche quali ottimizzazione bayesiana, ricerca casuale e ricerca a griglia
- Neural Architecture Search (NAS) per cercare automaticamente architetture di rete ottimali
- AutoML, che come descritto in Sezione 6.5, automatizza la pipeline ML.

6.11 Conclusione

In sintesi, la selezione del framework di machine learning ottimale richiede una valutazione approfondita di varie opzioni in base a criteri quali usabilità, supporto della community, prestazioni, compatibilità hardware e capacità di conversione del modello. Non esiste una soluzione adatta a tutti, poiché il framework giusto dipende da vincoli e casi d'uso specifici.
Abbiamo prima introdotto la necessità di framework di apprendimento automatico come TensorFlow e PyTorch. Questi framework offrono funzionalità quali tensori per la gestione di dati multidimensionali, grafi computazionali per la definizione e l'ottimizzazione delle operazioni del modello e una suite di strumenti tra cui funzioni di perdita, ottimizzatori e caricatori di dati che semplificano lo sviluppo del modello.
Le funzionalità avanzate migliorano ulteriormente l'usabilità di questi framework, consentendo attività come la messa a punto di grandi modelli pre-addestrati e la facilitazione del "federated learning". Queste funzionalità sono fondamentali per sviluppare modelli di apprendimento automatico sofisticati in modo efficiente.
I framework di intelligenza artificiale embedded o TinyML, come TensorFlow Lite Micro, forniscono strumenti specializzati per distribuire modelli su piattaforme con risorse limitate. TensorFlow Lite Micro, ad esempio, offre strumenti di ottimizzazione completi, tra cui la mappatura della quantizzazione e le ottimizzazioni del kernel, per garantire prestazioni elevate su piattaforme basate su microcontrollori come i processori Arm Cortex-M e RISC-V. I framework creati appositamente per hardware specializzato come CMSIS-NN su processori Cortex-M possono massimizzare ulteriormente le prestazioni ma sacrificare la portabilità. I framework integrati dei fornitori di processori adattano lo stack alle loro architetture, liberando il pieno potenziale dei loro chip ma ci si blocca nel loro ecosistema.
In definitiva, la scelta del framework giusto implica la ricerca della migliore corrispondenza tra le sue capacità e i requisiti della piattaforma target. Ciò richiede un bilanciamento tra esigenze di prestazioni, vincoli hardware, complessità del modello e altri fattori. Una valutazione approfondita dei modelli e dei casi d'uso previsti e la valutazione delle opzioni

rispetto alle metriche chiave guideranno gli sviluppatori nella selezione del framework ideale per le loro applicazioni di machine learning.

6.12 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Frameworks overview.
- Embedded systems software.
- Inference engines: TF vs. TFLite.
- TF flavors: TF vs. TFLite vs. TFLite Micro.
- TFLite Micro:
  - TFLite Micro Big Picture.
  - TFLite Micro Interpreter.
  - TFLite Micro Model Format.
  - TFLite Micro Memory Allocation.
  - TFLite Micro NN Operations.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 6.1
- Esercizio 6.2
- Esercizio 6.3

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

7. Addestramento dell'IA

DALL·E 3 Prompt: Un'illustrazione per l'addestramento dell'IA, raffigurante una rete neurale con neuroni che vengono riparati e attivati. La scena include una vasta rete di neuroni, ognuno dei quali si illumina e si attiva per rappresentare attività e apprendimento. Tra questi neuroni, piccole figure che ricordano ingegneri e scienziati lavorano attivamente, riparando e modificando i neuroni. Questi lavoratori in miniatura simboleggiano il processo di addestramento della rete, regolando pesi e bias per ottenere la convergenza. L'intera scena è una metafora visiva dello sforzo intricato e collaborativo coinvolto nell'addestramento dell'IA, con i lavoratori che rappresentano l'ottimizzazione e l'apprendimento continui all'interno di una rete neurale. Lo sfondo è una serie complessa di neuroni interconnessi, che creano un senso di profondità e complessità.

Il training è fondamentale per sviluppare sistemi di IA accurati e utili utilizzando tecniche di apprendimento automatico. A un livello elevato, il training implica l'inserimento di dati negli algoritmi di machine learning in modo che possano imparare pattern e fare previsioni. Tuttavia, il training efficace dei modelli richiede di affrontare vari problemi relativi a dati, algoritmi, ottimizzazione dei parametri del modello e abilitazione della generalizzazione. Questo capitolo esplorerà le sfumature e le considerazioni relative al training dei modelli di apprendimento automatico.

Obiettivi dell'Apprendimento

- Comprendere la matematica fondamentale delle reti neurali, tra cui le trasformazioni lineari, funzioni di attivazione, funzioni di perdita, backpropagation e ottimizzazione tramite "gradient descent" [discesa del gradiente].
- Scoprire come sfruttare efficacemente i dati per il training del modello tramite la suddivisione appropriata in set di addestramento, convalida e test per abilitare la generalizzazione.
- Scoprire vari algoritmi di ottimizzazione come lo "stochastic gradient descent" e adattamenti come momentum e Adam che accelerano l'addestramento.
- Scoprire tecniche di ottimizzazione e regolarizzazione degli iperparametri per migliorare la generalizzazione del modello riducendo l'overfitting.
- Scoprire strategie di inizializzazione del peso appropriate abbinate alle architetture del modello e alle scelte di attivazione che accelerano la convergenza.
- Identificare i colli di bottiglia posti da operazioni chiave come la moltiplicazione di matrici durante l'addestramento e il deployment.
- Scoprire come i miglioramenti hardware come GPU, TPU e acceleratori specializzati velocizzano le operazioni matematiche critiche per accelerare l'addestramento.
- Scoprire tecniche di parallelizzazione, sia parallelismo dei dati che del modello, per distribuire l'addestramento su più dispositivi e accelerare la produttività del sistema.

7.1 Introduzione

Il training è fondamentale per sviluppare sistemi di intelligenza artificiale accurati e utili tramite il machine learning. Il training crea un modello di apprendimento automatico che può essere generalizzato a dati nuovi e inediti anziché memorizzare gli esempi dell'addestramento. Ciò avviene inserendo dati di training in algoritmi che apprendono pattern da questi esempi regolando i parametri interni.

Gli algoritmi riducono al minimo una "funzione loss" [perdita], che confronta le loro previsioni sui dati di training con le etichette o le soluzioni note, guidando l'apprendimento. Un training efficace richiede spesso set di dati rappresentativi di alta qualità sufficientemente grandi da catturare la variabilità nei casi d'uso del mondo reale.

Richiede inoltre la scelta di un algoritmo adatto all'attività, che si tratti di una rete neurale per la visione artificiale, un algoritmo di apprendimento di rinforzo per il controllo robotico o un metodo basato su alberi per la previsione categoriale. È necessaria un'attenta messa a punto per la struttura del modello, come la profondità e la larghezza della rete neurale e i parametri di apprendimento come la dimensione del passo e la forza della regolarizzazione.

Sono importanti anche le tecniche per prevenire l'overfitting, come le penalità di regolarizzazione nonché la convalida con dati trattenuti. L'overfitting può verificarsi quando un modello si adatta troppo ai dati di training, non riuscendo a generalizzare con i nuovi dati. Ciò può accadere se il modello è troppo complesso o è stato addestrato troppo a lungo.

Per evitare l'overfitting, le tecniche di regolarizzazione possono aiutare a vincolare il modello. Un metodo di regolarizzazione consiste nell'aggiungere un termine di penalità alla funzione di perdita che scoraggia la complessità, come la norma L2 dei pesi. Questo penalizza i valori dei parametri elevati. Un'altra tecnica è il dropout, in cui una percentuale di neuroni viene impostata casualmente a zero durante l'addestramento. Ciò riduce il co-adattamento dei neuroni.

I metodi di validazione aiutano anche a rilevare ed evitare l'overfitting. Una parte dei dati di training viene tenuta fuori dal ciclo di training come un set di validazione. Il modello viene valutato su questi dati. Se l'errore di convalida aumenta mentre l'errore di training diminuisce, si verifica un overfitting. Il training può quindi essere interrotto in anticipo o regolarizzato in modo più forte. La regolarizzazione e la convalida consentono ai modelli di addestrarsi alla massima capacità senza overfitting [sovra-adattare] i dati di training.
Il training richiede notevoli risorse di elaborazione, in particolare per le reti neurali profonde (deep) utilizzate nella visione artificiale, nell'elaborazione del linguaggio naturale e in altre aree. Queste reti hanno milioni di pesi regolabili che devono essere regolati tramite un training esteso. I miglioramenti hardware e le tecniche di training distribuite hanno consentito di addestrare reti neurali sempre più grandi che possono raggiungere prestazioni di livello umano in alcune attività.
In sintesi, alcuni punti chiave sul training:

- **I Dati sono cruciali:** I modelli di machine learning apprendono dagli esempi nei dati di training. Dati più rappresentativi e di qualità elevata portano a migliori prestazioni del modello. I dati devono essere elaborati e formattati per il training.
- **Gli algoritmi imparano dai dati:** Diversi algoritmi (reti neurali, alberi decisionali, ecc.) hanno approcci diversi per trovare dei pattern nei dati. È importante scegliere l'algoritmo giusto per l'attività.
- **L'addestramento affina i parametri del modello:** L'addestramento del modello regola i parametri interni per trovare pattern nei dati. I modelli avanzati come le reti neurali hanno molti pesi regolabili. L'addestramento regola iterativamente i pesi per ridurre al minimo una funzione di perdita.
- **La generalizzazione è l'obiettivo:** Un modello che sovra-adatta i dati di addestramento non generalizzerà bene. Le tecniche di regolarizzazione (dropout, early stopping arresto anticipato, ecc.) riducono il sovra-adattamento. I dati di validazione vengono utilizzati per valutare la generalizzazione.
- **L'addestramento richiede risorse di elaborazione:** L'addestramento di modelli complessi richiede una notevole potenza di elaborazione e tempo. I miglioramenti hardware e il training distribuito su GPU/TPU hanno consentito dei progressi.

Guideremo attraverso questi dettagli nelle restanti sezioni. Comprendere come sfruttare in modo efficace dati, algoritmi, ottimizzazione dei parametri e generalizzazione attraverso il training è essenziale per sviluppare sistemi di intelligenza artificiale capaci e distribuibili che funzionino in modo robusto nel mondo reale.

7.2 Matematica delle Reti Neurali

Il deep learning ha rivoluzionato l'apprendimento automatico e l'intelligenza artificiale, consentendo ai computer di apprendere pattern complessi e prendere decisioni intelligenti. La rete neurale è al centro della rivoluzione del deep learning e, come discusso nella sezione 3, "Avvio al Deep Learning", è un pilastro in alcuni di questi progressi.
Le reti neurali sono costituite da semplici funzioni stratificate l'una sull'altra. Ogni layer acquisisce alcuni dati, esegue alcuni calcoli e li passa al layer successivo. Questi layer apprendono progressivamente funzionalità di alto livello utili per le attività che la rete è addestrata a svolgere. Ad esempio, in una rete addestrata per il riconoscimento delle immagini, il layer di input può acquisire valori di pixel, mentre i layer successivi possono rilevare forme semplici come i bordi. I layer successivi possono rilevare forme più complesse come nasi, occhi, ecc. Il layer di output finale classifica l'immagine nel suo complesso.
La rete, in una rete neurale, si riferisce al modo in cui questi layer sono connessi. L'output di ogni layer è considerato un set di neuroni, che sono collegati ai neuroni nei layer successivi,

formando una "rete". Il modo in cui questi neuroni interagiscono è determinato dai pesi tra di loro, che modellano le forze sinaptiche simili a quelle di un neurone del cervello. La rete neurale viene addestrata regolando questi pesi. Concretamente, i pesi vengono inizialmente impostati in modo casuale, quindi viene immesso l'input, l'output viene confrontato con il risultato desiderato e, infine, i pesi vengono modificati per migliorare la rete. Questo processo viene ripetuto finché la rete non riduce al minimo in modo affidabile la perdita (loss), indicando di aver appreso i pattern nei dati.
Come viene definito matematicamente questo processo? Formalmente, le reti neurali sono modelli matematici costituiti da operazioni lineari e non lineari alternate, parametrizzate da un set di pesi apprendibili che vengono addestrati per minimizzare una qualche funzione di perdita (loss). Questa funzione di perdita misura quanto è buono il nostro modello per quanto riguarda l'adattamento dei nostri dati di addestramento e produce un valore numerico quando viene valutato sul nostro modello rispetto ai dati di addestramento. L'addestramento delle reti neurali comporta la valutazione ripetuta della funzione di perdita su molti dati diversi per misurare quanto è buono il nostro modello, quindi la modifica continua dei pesi del nostro modello utilizzando la backpropagation in modo che la perdita diminuisca, ottimizzando infine il modello per adattarlo ai nostri dati.

7.2.1 Notazione delle Reti Neurali

Il nucleo di una rete neurale può essere visto come una sequenza di operazioni lineari e non lineari alternate, come mostrato in Figura 7.1.

Figura 7.1: Diagramma della rete neurale. Fonte: astroML.

Le reti neurali sono strutturate con layer [strati] di neuroni collegati da pesi (che rappresentano operazioni lineari) e funzioni di attivazione (che rappresentano operazioni non lineari). Esaminando la figura, vediamo come le informazioni fluiscono attraverso la rete, partendo dal layer di input, passando attraverso uno o più layer nascosti, e infine raggiungendo il layer di output. Ogni connessione tra neuroni rappresenta un peso, mentre ciascun neurone applica tipicamente una funzione di attivazione non lineare ai suoi input.
La rete neurale funziona prendendo un vettore di input x_i e passandolo attraverso una serie di layer, ognuno dei quali esegue operazioni lineari e non lineari. L'output della rete a ogni layer A_j può essere rappresentato come:

$$A_j = f\left(\sum_{i=1}^{N} w_{ij}\, x_i\right)$$

Dove:

- N - Il numero totale di feature di input.
- x_i - La singola feature di input, dove i varia da 1 a N.
- w_{ij} - I pesi che collegano il neurone i in uno layer al neurone j nel layer successivo, che vengono aggiustati durante l'addestramento.
- $f(\theta)$ - La funzione di attivazione non lineare applicata a ogni layer (ad esempio, ReLU, softmax, ecc.).
- A_j - L'output della rete neurale a ogni layer j, dove j indica il numero del layer.

Nel contesto di Figura 7.1, x_1, x_2, x_3, x_4, e x_5 rappresentano le caratteristiche di input. Ogni neurone di input x_i corrisponde a una feature dei dati di input. Le frecce dal layer di input al layer nascosto indicano le connessioni tra i neuroni di input e i neuroni nascosti, con ogni connessione associata a un peso w_{ij}.

Il layer nascosto è costituito dai neuroni a_1, a_2, a_3, e a_4, ognuno dei quali riceve input da tutti i neuroni nello layer di input. I pesi w_{ij} collegano i neuroni di input ai neuroni nascosti. Ad esempio, w_{11} è il peso che collega l'input x_1 al neurone nascosto a_1.

Il numero di nodi in ogni layer e il numero totale di layer insieme definiscono l'architettura della rete neurale. Nel primo layer (layer di input), il numero di nodi corrisponde alla dimensionalità dei dati di input, mentre nell'ultimo layer (layer di output), il numero di nodi corrisponde alla dimensionalità dell'output. Il numero di nodi nei layer intermedi può essere impostato arbitrariamente, consentendo flessibilità nella progettazione dell'architettura di rete.

I pesi, che determinano il modo in cui ogni layer della rete neurale interagisce con gli altri, sono matrici di numeri reali. Inoltre, ogni layer in genere include un vettore di bias [polarizzazione], ma qui lo ignoriamo per semplicità. La matrice dei pesi W_j che collega il layer $j-1$ al layer j ha le dimensioni:

$$W_j \in \mathbb{R}^{d_j \times d_{j-1}}$$

dove d_j è il numero di nodi nel layer j e d_{j-1} è il numero di nodi nel layer precedente $j-1$.

L'output finale y_k della rete si ottiene applicando un'altra funzione di attivazione $g(\theta)$ alla somma ponderata degli output del layer nascosto:

$$y = g\left(\sum_{j=1}^{M} w_{jk}\, A_j\right)$$

Dove:

- M - Il numero di neuroni nascosti nel layer finale prima dell'output.
- w_{jk} - Il peso tra il neurone nascosto a_j e il neurone di output y_k.
- $g(\theta)$ - La funzione di attivazione applicata alla somma ponderata degli output del layer nascosto.

La nostra rete neurale, come definita, esegue una sequenza di operazioni lineari e non lineari sui dati di input (x_i) per ottenere previsioni (y_i), che si spera siano una buona risposta a ciò che vogliamo che la rete neurale faccia sull'input (ad esempio, classificare se l'immagine di input è un gatto o meno). La nostra rete neurale può quindi essere rappresentata succintamente come una funzione N che accetta un input $x \in \mathbb{R}^{d_0}$ parametrizzato da $W_1, \ldots, W_n$ e produce l'output finale y:

$$y = N(x; W_1, \ldots, W_n) \quad \text{where } A_0 = x$$

Questa equazione indica che la rete inizia con l'input $A_0 = x$ e calcola iterativamente A_j a ogni layer utilizzando i parametri W_j fino a produrre l'output finale y al layer di output. Successivamente vedremo come valutare questa rete neurale rispetto ai dati di addestramento introducendo una funzione di perdita.

> ⓘ Nota
>
> Perché sono necessarie le operazioni non lineari? Se avessimo solo layer lineari, l'intera rete sarebbe equivalente a un singolo layer lineare costituito dal prodotto degli operatori lineari. Quindi, le funzioni non lineari svolgono un ruolo chiave nella potenza delle reti neurali poiché migliorano la capacità della rete neurale di adattare le funzioni.

> ⓘ Nota
>
> Anche le convoluzioni sono operatori lineari e possono essere convertite in una moltiplicazione di matrici.

7.2.2 Funzione Loss come Misura della Bontà di Adattamento Rispetto ai Dati di Addestramento

Dopo aver definito la nostra rete neurale, ci vengono forniti alcuni dati di addestramento, ovvero un set di punti (x_j, y_j) per $j = 1 \to M$, dove M è il numero totale di campioni nel set di dati e j indicizza ogni campione. Vogliamo valutare quanto è buona la nostra rete neurale nell'adattare questi dati. Per fare ciò, introduciamo una funzione di perdita, ovvero una funzione che prende l'output della rete neurale su un particolare punto dati $\hat{y}_j = N(x_j; W_1, \ldots, W_n)$ e lo confronta con la "etichetta" di quel particolare dato (il corrispondente y_j) e restituisce un singolo scalare numerico (ovvero un numero reale) che rappresenta quanto è "bene" la rete neurale adatta quel particolare dato; la misura finale di quanto è buona la rete neurale sull'intero set di dati è quindi solo la media delle perdite su tutti i dati.
Esistono molti tipi diversi di funzioni di perdita; ad esempio, nel caso della classificazione delle immagini, potremmo usare la funzione di "cross-entropy loss" [perdita di entropia incrociata], che ci dice quanto bene si confrontano due vettori che rappresentano le previsioni di classificazione (ad esempio, se la nostra previsione prevede che un'immagine sia più probabilmente un cane, ma l'etichetta dice che è un gatto, restituirà una "perdita" elevata, che indica un cattivo adattamento).
Matematicamente, una funzione di perdita è una funzione che prende due vettori con valori reali, uno che rappresenta gli output previsti della rete neurale e l'altro che rappresenta le etichette vere, e restituisce un singolo scalare numerico che rappresenta l'errore o la "perdita".

$$L: \mathbb{R}^{d_n} \times \mathbb{R}^{d_n} \to \mathbb{R}$$

Per un singolo esempio di training, la perdita è data da:

$$L(N(x_j; W_1, \ldots, W_n), y_j)$$

dove $\hat{y}_j = N(x_j; W_1, \ldots, W_n)$ è l'output previsto della rete neurale per l'input x_j, and y_j è la vera etichetta.
La perdita totale nell'intero set di dati, L_{full}, viene quindi calcolata come la perdita media in tutti i dati di training:

Funzione di Perdita per l'Ottimizzazione del Modello di Rete Neurale su Dataset

$$L_{full} = \frac{1}{M}\sum_{j=1}^{M} L\left(N\left(x_j; W_1, \ldots W_n\right), y_j\right)$$

7.2.3 Addestramento di Reti Neurali con Discesa del Gradiente

Ora che possiamo misurare quanto bene la nostra rete si adatta ai dati di training, possiamo ottimizzare i pesi della rete neurale per ridurre al minimo questa perdita. In questo contesto, stiamo denotando W_i come pesi per ogni layer i nella rete. Ad alto livello, modifichiamo i parametri delle matrici a valori reali W_i per ridurre al minimo la funzione di perdita L_{full}. Nel complesso, il nostro obiettivo matematico è

Obiettivo dell'Addestramento della Rete Neurale

$$min_{W_1,\ldots,W_n} L_{full}$$
$$= min_{W_1,\ldots,W_n} \frac{1}{M}\sum_{j=1}^{M} L\left(N\left(x_j; W_1, \ldots W_n\right), y_j\right)$$

Quindi, come ottimizziamo questo obiettivo? Ricordiamo dal calcolo che la minimizzazione di una funzione può essere eseguita prendendo la derivata della funzione relativa ai parametri di input e modificando i parametri nella direzione del gradiente. Questa tecnica è chiamata a "discesa del gradiente" e concretamente comporta il calcolo della derivata della funzione di perdita L_{full} relativa a $W_1, \ldots, W_n$ per ottenere un gradiente per questi parametri per fare un passo avanti, poi aggiornare questi parametri nella direzione del gradiente. Quindi, possiamo addestrare la nostra rete neurale utilizzando la discesa del gradiente, che applica ripetutamente la regola di aggiornamento.

Regola di Aggiornamento della Discesa del Gradiente

$$W_i := W_i - \lambda \frac{\partial L_{full}}{\partial W_i} \text{ for } i = 1..n$$

> ⓘ Nota
>
> In pratica, il gradiente viene calcolato su un mini-batch di punti dati per migliorare l'efficienza computazionale. Questo processo è chiamato "discesa del gradiente stocastico" o "discesa del gradiente batch".

Dove λ è la dimensione del passo o il tasso di apprendimento delle nostre modifiche, nell'addestramento della nostra rete neurale, eseguiamo ripetutamente il passaggio precedente fino alla convergenza, o quando la perdita non diminuisce più. Figura 7.2 illustra questo processo: vogliamo raggiungere il punto minimo, il che si ottiene seguendo il gradiente (come illustrato con le frecce blu nella figura). Questo precedente approccio è noto come discesa del gradiente completa poiché stiamo calcolando la derivata relativa a tutti i dati di addestramento e solo dopo eseguiamo un singolo passaggio del gradiente; un approccio più efficiente è quello di calcolare il gradiente relativo solo a un batch casuale di dati e poi eseguire un passaggio, un processo noto come discesa del gradiente batch o discesa del gradiente stocastica (Robbins e Monro 1951), che è più efficiente poiché ora eseguiamo molti più passi per passaggio di tutti i dati di addestramento. Successivamente, tratteremo la matematica alla base del calcolo del gradiente della funzione di perdita relativa a W_i, un processo noto come backpropagation.

Figura 7.2: Discesa del gradiente. Fonte: Towards Data Science.

7.2.4 Backpropagation

L'addestramento delle reti neurali comporta ripetute applicazioni dell'algoritmo di discesa del gradiente, che prevede il calcolo della derivata della funzione di perdita rispetto alle W_i. Come calcoliamo la derivata della perdita relativa alle W_i, dato che le W_i sono funzioni annidate l'una dell'altra in una rete neurale profonda? Il trucco è sfruttare la **regola della catena:** possiamo calcolare la derivata della perdita relativa alle W_i applicando ripetutamente la regola della catena in un processo completo noto come backpropagation. In particolare, possiamo calcolare i gradienti calcolando la derivata della perdita relativa agli output dell'ultimo layer, poi usarla progressivamente per calcolare la derivata della perdita relativa a ciascun layer precedente a quello di input. Questo processo inizia dalla fine della rete (il layer più vicino all'output) e procede all'indietro, e quindi prende il nome di backpropagation.

Analizziamolo. Possiamo calcolare la derivata della perdita relativa agli *output di ciascun layer della rete neurale* utilizzando applicazioni ripetute della regola della catena.

$$\frac{\partial L_{full}}{\partial L_n} = \frac{\partial A_n}{\partial L_n}\frac{\partial L_{full}}{\partial A_n}$$

$$\frac{\partial L_{full}}{\partial L_{n-1}} = \frac{\partial A_{n-1}}{\partial L_{n-1}}\frac{\partial L_n}{\partial A_{n-1}}\frac{\partial A_n}{\partial L_n}\frac{\partial L_{full}}{\partial A_n}$$

o più in generale

$$\frac{\partial L_{full}}{\partial L_i} = \frac{\partial A_i}{\partial L_i}\frac{\partial L_{i+1}}{\partial A_i}\cdots\frac{\partial A_n}{\partial L_n}\frac{\partial L_{full}}{\partial A_n}$$

Nota

In quale ordine dovremmo eseguire questo calcolo? Da una prospettiva computazionale, è preferibile eseguire i calcoli dalla fine alla parte frontale. (ad esempio: prima si calcola $\frac{\partial L_{full}}{\partial A_n}$, poi i termini precedenti, anziché iniziare dal centro) poiché ciò evita di materializzare e calcolare grandi jacobiani. Questo perché $\frac{\partial L_{full}}{\partial A_n}$ è un vettore; quindi, qualsiasi operazione di matrice che include questo termine ha un output che è compresso per essere un vettore. Quindi, eseguire il calcolo dalla fine evita grandi moltiplicazioni matrice-matrice assicurando che i prodotti intermedi siano vettori.

Nota

Nella nostra notazione, assumiamo che le attivazioni intermedie A_i siano vettori *colonna*, anziché vettori *riga*, quindi la regola della catena è $\frac{\partial L}{\partial L_i} = \frac{\partial L_{i+1}}{\partial L_i} \dots \frac{\partial L}{\partial L_n}$ piuttosto che $\frac{\partial L}{\partial L_i} = \frac{\partial L}{\partial L_n} \dots \frac{\partial L_{i+1}}{\partial L_i}$

Dopo aver calcolato la derivata della perdita relativa all'*output di ogni layer*, possiamo facilmente ottenere la derivata della perdita relativa ai *parametri*, utilizzando di nuovo la regola della catena:

$$\frac{\partial L_{full}}{W_i} = \frac{\partial L_i}{\partial W_i} \frac{\partial L_{full}}{\partial L_i}$$

Ed è in definitiva così che le derivate dei pesi dei layer vengono calcolate usando la backpropagation! Come appare concretamente in un esempio specifico? Di seguito, esaminiamo un esempio specifico di una semplice rete neurale a 2 layer su un'attività di regressione usando una funzione di perdita MSE con input a 100 dimensioni e uno layer nascosto a 30 dimensioni:

Esempio di backpropagation

Supponiamo di avere una rete neurale a due layer

$$L_1 = W_1 A_0$$
$$A_1 = ReLU(L_1)$$
$$L_2 = W_2 A_1$$
$$A_2 = ReLU(L_2)$$
$$NN(x) = \text{Let } A_0 = x \text{ then output } A_2$$

dove $W_1 \in \mathbb{R}^{30\times100}$ e $W_2 \in \mathbb{R}^{1\times30}$. Inoltre, supponiamo di utilizzare la funzione di perdita MSE:

$$L(x, y) = (x - y)^2$$

Vogliamo calcolare

$$\frac{\partial L(NN(x), y)}{\partial W_i} \text{ for } i = 1,2$$

Notare quanto segue:

$$\frac{\partial L(x, y)}{\partial x} = 2 \times (x - y)$$

$$\frac{\partial ReLU(x)}{\partial x}\delta = \begin{Bmatrix} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x \geq 0 \end{Bmatrix} \odot \delta$$

$$\frac{\partial WA}{\partial A}\delta = W^T\delta$$

$$\frac{\partial WA}{\partial W}\delta = \delta A^T$$

Quindi abbiamo

$$\frac{\partial L(NN(x), y)}{\partial W_2} = \frac{\partial L_2}{\partial W_2}\frac{\partial A_2}{\partial L_2}\frac{\partial L(NN(x), y)}{\partial A_2}$$
$$= \big(2L(NN(x) - y) \odot ReLU'(L_2)\big)A_1^T$$

e

$$\frac{\partial L(NN(x), y)}{\partial W_1} = \frac{\partial L_1}{\partial W_1}\frac{\partial A_1}{\partial L_1}\frac{\partial L_2}{\partial A_1}\frac{\partial A_2}{\partial L_2}\frac{\partial L(NN(x), y)}{\partial A_2}$$
$$= [ReLU'(L_1) \odot (W_2^T[2L(NN(x) - y) \odot ReLU'(L_2)])]A_0^T$$

Consiglio

Ricontrollare il lavoro assicurandosi che le forme siano corrette!

- Tutti i prodotti di Hadamard ($\odot$) dovrebbero operare su tensori della stessa forma
- Tutte le moltiplicazioni di matrici dovrebbero operare su matrici che condividono una dimensione comune (ad esempio, m per n, n per k)
- Tutti i gradienti relativi ai pesi dovrebbero avere la stessa forma delle stesse matrici dei pesi

L'intero processo di backpropagation può essere complesso, specialmente per reti molto profonde. Fortunatamente, framework di machine learning come PyTorch supportano la differenziazione automatica, che esegue la backpropagation. In questi framework, dobbiamo semplicemente specificare il passaggio in avanti e le derivate ci verranno calcolate automaticamente. Tuttavia, è utile comprendere il processo teorico che avviene internamente in questi framework di apprendimento automatico.

Nota

Come visto sopra, le attivazioni intermedie A_i vengono riutilizzate nella backpropagation. Per migliorare le prestazioni, queste attivazioni vengono memorizzate nella cache dal passaggio in avanti per evitare di essere ricalcolate. Tuttavia, le attivazioni devono essere mantenute in memoria tra i passaggi in avanti e indietro, il che comporta un maggiore utilizzo della memoria. Se la rete e le dimensioni del batch sono grandi, ciò potrebbe causare problemi di memoria. Analogamente, le derivate rispetto agli output di ogni layer vengono memorizzate nella cache per evitare il ricalcolo.

Esercizio 7.1: Reti Neurali con Backpropagation e Discesa del Gradiente

Scoprire la matematica dietro le potenti reti neurali! Il deep learning potrebbe sembrare magico, ma è radicato nei principi matematici. In questo capitolo, abbiamo scomposto la notazione delle reti neurali, le funzioni di perdita e la potente tecnica della backpropagation. Ora, prepariamoci a implementare questa teoria con questi notebook Colab. Immergersi nel cuore di come le reti neurali apprendono. Si vedrà la matematica dietro la backpropagation e la discesa del gradiente, aggiornando quei pesi passo dopo passo.

Open in Colab

7.3 Grafi del Calcolo Differenziabili

In generale, la discesa del gradiente stocastico mediante backpropagation può essere eseguita su qualsiasi grafo computazionale che un utente può definire, a condizione che le operazioni del calcolo siano differenziabili. Pertanto, le librerie generiche di deep learning come PyTorch e Tensorflow consentono agli utenti di specificare il loro processo computazionale (ad esempio, reti neurali) come grafo computazionale. La backpropagation viene eseguita automaticamente tramite differenziazione automatica quando la discesa del gradiente stocastico viene eseguita su questi grafi computazionali. Inquadrare l'addestramento dell'IA come un problema di ottimizzazione su grafi di calcolo differenziabili è un modo generale per comprendere cosa sta accadendo internamente con i sistemi di deep learning.

La struttura raffigurata in Figura 7.3 mostra un segmento di un grafo computazionale differenziabile. In questo grafo, l'input 'x' viene elaborato tramite una serie di operazioni: viene prima moltiplicato per una matrice di pesi 'W' (MatMul), poi aggiunto a un bias 'b' (Add) e infine passato a una funzione di attivazione, Rectified Linear Unit (ReLU). Questa sequenza di operazioni ci fornisce l'output C. La natura differenziabile del grafo significa che ogni operazione ha un gradiente ben definito. La differenziazione automatica, come implementata nei framework ML, sfrutta questa proprietà per calcolare in modo efficiente i gradienti della perdita rispetto a ciascun parametro nella rete (ad esempio, 'W' e 'b').

Figura 7.3: Grafo Computazionale. Fonte: TensorFlow.

7.4 Dati di Training

Per consentire un training efficace della rete neurale, i dati disponibili devono essere suddivisi in set di training, di validazione e di test. Il set di training viene utilizzato per addestrare i parametri del modello. Il set di validazione valuta il modello durante il training per ottimizzare gli iperparametri e prevenire l'overfitting. Il set di test fornisce una valutazione finale imparziale delle prestazioni del modello addestrato.
Mantenere chiare suddivisioni tra training, validation e test con dati rappresentativi è fondamentale per addestrare, ottimizzare e valutare correttamente i modelli per ottenere le migliori prestazioni nel mondo reale. A tal fine, scopriremo le insidie o gli errori comuni che le persone commettono quando creano queste suddivisioni dei dati.
Tabella 7.1 confronta le differenze tra le suddivisioni dei dati di training, validazione e test:

Tabella 7.1: Confronto tra suddivisioni di dati di training, validazione e test.

Suddivisione	Scopo	Dimensioni tipiche
Set di addestramento	Addestrare i parametri del modello	60-80% dei dati totali
Set di validazione	Valutare il modello durante l'addestramento per ottimizzare gli iperparametri e prevenire l'overfitting	~20% dei dati totali
Set di test	Fornire una valutazione imparziale del modello finale addestrato	~20% dei dati totali

7.4.1 Suddivisioni di Dataset

7.4.1.1 Set di Training

Il set di training viene utilizzato per addestrare il modello. È il sottoinsieme più grande, in genere il 60-80% dei dati totali. Il modello vede e impara dai dati di addestramento per fare previsioni. È necessario un set di training sufficientemente grande e rappresentativo affinché il modello apprenda efficacemente i pattern sottostanti.

7.4.1.2 Set di Validazione

Il set di validazione valuta il modello durante l'addestramento, in genere dopo ogni epoca. Solitamente, il 20% dei dati viene assegnato a questo set. Il modello non impara né aggiorna i suoi parametri in base ai dati di validazione. Vengono usati per ottimizzare gli iperparametri e apportare altre modifiche per migliorare l'addestramento. Il monitoraggio di metriche come perdita e accuratezza sul set di validazione impedisce l'overfitting solo sui dati di addestramento.

7.4.1.3 Set di Test

Il set di test agisce come un dataset che il modello non ha visto durante l'addestramento. Viene utilizzato per fornire una valutazione imparziale del modello addestrato finale. In genere, il 20% dei dati è riservato ai test. Mantenere un set di test "hold-out" [esterno] è fondamentale per ottenere una stima accurata di come il modello addestrato si comporterebbe su dati non ancora visti del mondo reale. La mancanza di dati dal set di test deve essere evitata a tutti i costi.
Le proporzioni relative dei set di training, validazione e test possono variare in base alle dimensioni dei dati e all'applicazione. Tuttavia, seguire le linee guida generali per una suddivisione 60/20/20 è un buon punto di partenza. Un'attenta suddivisione dei dati garantisce che i modelli siano adeguatamente addestrati, ottimizzati e valutati per ottenere le prestazioni migliori.

Video 7.1 spiega come suddividere correttamente il dataset in set di training, validazione e test, assicurando un processo di training ottimale.

Video 7.1: Train/Dev/Test Sets

https://www.youtube.com/watch?v=1waHlpKiNyY

7.4.2 Errori e Insidie Comuni

7.4.2.1 Dati di Training Insufficienti

Assegnare troppo pochi dati al set di training è un errore comune quando si suddividono i dati, il che può avere un impatto significativo sulle prestazioni del modello. Se il set di training è troppo piccolo, il modello non avrà campioni sufficienti per apprendere in modo efficace i veri pattern nei dati. Ciò comporta un'elevata varianza e impedisce al modello di generalizzare bene ai nuovi dati.

Ad esempio, se si addestra un modello di classificazione delle immagini per riconoscere cifre scritte a mano, fornire solo 10 o 20 immagini per classe di cifre sarebbe del tutto inadeguato. Il modello avrebbe bisogno di più esempi per catturare le ampie varianze negli stili di scrittura, rotazioni, larghezze dei tratti e altre varianti.

Come regola generale, la dimensione del training set dovrebbe essere di almeno centinaia o migliaia di esempi affinché la maggior parte degli algoritmi di apprendimento automatico funzioni in modo efficace. A causa dell'elevato numero di parametri, il set di training spesso deve essere di decine o centinaia di migliaia per le reti neurali profonde, in particolare quelle che utilizzano layer convoluzionali.

Dati di training insufficienti si manifestano in genere in sintomi quali alti tassi di errore su set di validazione/test, bassa accuratezza del modello, alta varianza e overfitting su campioni di set di training di piccole dimensioni. La soluzione è raccogliere più dati di training di qualità. Le tecniche di data augmentation possono anche aiutare ad aumentare virtualmente le dimensioni dei dati di training per immagini, audio, ecc.

È importante considerare attentamente la complessità del modello e la difficoltà del problema quando si assegnano i campioni di training per garantire che siano disponibili dati sufficienti affinché il modello possa apprendere correttamente. Si consiglia inoltre di seguire le linee guida sulle dimensioni minime dei set di training per diversi algoritmi. Sono necessari più dati di training per mantenere il successo complessivo di qualsiasi applicazione di machine learning.

Si consideri Figura 7.4 dove proviamo a classificare/suddividere i dati in due categorie (qui, per colore): a sinistra, l'overfitting è rappresentato da un modello che ha appreso troppo bene le sfumature nei dati di training (o il set di dati era troppo piccolo o abbiamo eseguito il modello per troppo tempo), facendo sì che segua il rumore insieme al segnale, come indicato dalle eccessive curve della linea. Il lato destro mostra l'underfitting, dove la semplicità del modello gli impedisce di catturare la struttura sottostante del dataset, con conseguente linea che non si adatta bene ai dati. Il grafico centrale rappresenta un adattamento ideale, dove il modello bilancia bene tra generalizzazione e adattamento, catturando la tendenza principale dei dati senza essere influenzato da valori anomali. Sebbene il modello non sia un adattamento perfetto (manca di alcuni punti), ci interessa di più la sua capacità di riconoscere pattern generali piuttosto che valori anomali idiosincratici.

Figura 7.4: Adattamento dei dati: overfitting, right fit e underfitting. Fonte: MathWorks.

Figura 7.5 il processo di adattamento dei dati nel tempo. Durante l'addestramento, cerchiamo il "punto ottimale" tra underfitting e overfitting. Inizialmente, quando il modello non ha avuto abbastanza tempo per apprendere i pattern nei dati, ci troviamo nella zona di underfitting, indicata da alti tassi di errore sul set di convalida (da ricordare che il modello è addestrato sul set di addestramento e testiamo la sua generalizzabilità sul set di convalida o sui dati che non ha mai visto prima). A un certo punto, raggiungiamo un minimo globale per i tassi di errore e idealmente vogliamo interrompere l'addestramento lì. Se continuiamo l'addestramento, il modello inizierà a "memorizzare" o a conoscere i dati troppo bene, tanto che il tasso di errore inizierà a risalire, poiché il modello non riuscirà a generalizzare a dati che non ha mai visto prima.

Figura 7.5: Adattamento dei dati nel tempo. Fonte: IBM.

Il Video 7.2 fornisce una panoramica di bias e varianza e la relazione tra i due concetti e l'accuratezza del modello.

Video 7.2: Bias/Varianza
https://www.youtube.com/watch?v=SjQyLhQIXSM

7.4.2.2 Perdita di Dati Tra Set

Il "data leakage" [perdita di dati] si riferisce al trasferimento involontario di informazioni tra i set di training, convalida e test. Ciò viola il presupposto fondamentale che le divisioni siano

reciprocamente esclusive. La perdita di dati porta a risultati di valutazione seriamente compromessi e metriche di prestazioni gonfiate.
Un modo comune in cui si verifica la perdita di dati è se alcuni campioni del set di test vengono inavvertitamente inclusi nei dati di training. Quando si valuta il set di test, il modello ha già visto alcuni dati, il che fornisce punteggi eccessivamente ottimistici. Ad esempio, se il 2% dei dati di test trapelano nel set di training di un classificatore binario, può comportare un aumento della precisione fino al 20%!
Se le divisioni dei dati non vengono eseguite con attenzione, possono verificarsi forme di perdita più sottili. Se le divisioni non vengono randomizzate e mescolate correttamente, i campioni che sono vicini tra loro nel set di dati potrebbero finire nella stessa divisione, portando a distorsioni della distribuzione. Ciò crea una fuga di informazioni basata sulla prossimità nel set di dati.
Un altro caso è quando i set di dati hanno campioni collegati, intrinsecamente connessi, come grafici, reti o dati di serie temporali. La suddivisione "ingenua" può isolare nodi o intervalli di tempo connessi in set diversi. I modelli possono fare ipotesi non valide basate su informazioni parziali.
Per prevenire la perdita di dati è necessario creare una solida separazione tra le suddivisioni: nessun campione dovrebbe esistere in più di una suddivisione. Il mescolamento e la suddivisione randomizzata aiutano a creare divisioni robuste. Le tecniche di "cross-validation" [validazione incrociata] possono essere utilizzate per una valutazione più rigorosa. Rilevare la perdita è difficile, ma i segnali rivelatori includono modelli che funzionano molto meglio sui dati di test rispetto a quelli di validazione.
La perdita di dati compromette gravemente la validità della valutazione perché il modello ha già visto parzialmente i dati di test. Nessuna quantità di messa a punto o architetture complesse può sostituire le suddivisioni nette dei dati. È meglio essere prudenti e creare una separazione completa tra le suddivisioni per evitare questo errore fondamentale nelle pipeline di machine learning.

7.4.2.3 Set di Validazione Piccolo o Non Rappresentativo

Il set di validazione viene utilizzato per valutare le prestazioni del modello durante l'addestramento e per ottimizzare gli iperparametri. Per valutazioni affidabili e stabili, il set di validazione deve essere sufficientemente ampio e rappresentativo della distribuzione dei dati reali. Tuttavia, ciò può rendere più impegnativa la selezione e l'ottimizzazione del modello.
Ad esempio, se il set di validazione contiene solo 100 campioni, le metriche calcolate avranno un'elevata varianza. A causa del rumore, l'accuratezza può variare fino al 5-10% tra le epoche. Questo rende difficile sapere se un calo nell'accuratezza della validazione è dovuto a un overfitting o a una varianza naturale. Con un set di validazione più ampio, diciamo 1000 campioni, le metriche saranno molto più stabili.
Inoltre, se il set di validazione non è rappresentativo, forse mancano alcune sottoclassi, la capacità stimata del modello potrebbe essere gonfiata. Ciò potrebbe portare a scelte di iperparametri scadenti o a interruzioni premature dell'addestramento. I modelli selezionati in base a tali set di validazione distorti non si generalizzano bene ai dati reali.
Una buona regola pratica è che la dimensione del set di convalida dovrebbe essere di almeno diverse centinaia di campioni e fino al 10-20% del set di addestramento, lasciando comunque campioni sufficienti per l'addestramento. Le divisioni dovrebbero anche essere stratificate, il che significa che le proporzioni di classe nel set di validazione dovrebbero corrispondere a quelle nel set di dati completo, soprattutto se si lavora con set di dati sbilanciati. Un set di validazione più ampio che rappresenti le caratteristiche dei dati originali è essenziale per una corretta selezione e messa a punto del modello.

7.4.2.4 Riutilizzo del Set di Test Più Volte

Il set di test è progettato per fornire una valutazione imparziale del modello completamente addestrato solo una volta alla fine del processo di sviluppo del modello. Riutilizzare il set di test più volte durante lo sviluppo per la valutazione del modello, la messa a punto degli iperparametri, la selezione del modello, ecc., può causare un overfitting sui dati di test. Invece, si deve riservare il set di test per una valutazione finale del modello completamente addestrato, trattandolo come una scatola nera per simularne le prestazioni su dati reali. Questo approccio fornisce metriche affidabili per determinare se il modello è pronto per la distribuzione in produzione.

Se il set di test viene riutilizzato come parte del processo di validazione, il modello potrebbe iniziare a vedere e imparare dai campioni di test. Questo, insieme all'ottimizzazione intenzionale o meno delle prestazioni del modello sul set di test, può gonfiare artificialmente metriche come l'accuratezza.

Ad esempio, supponiamo che il set di test venga utilizzato ripetutamente per la selezione del modello su 5 architetture. In tal caso, il modello potrebbe raggiungere il 99% di accuratezza del test memorizzando i campioni anziché apprendere pattern generalizzabili. Tuttavia, quando implementati nel mondo reale, l'accuratezza dei nuovi dati potrebbe scendere del 60%.

La prassi migliore è interagire con il set di test solo una volta alla fine per segnalare metriche imparziali su come il modello finale ottimizzato si comporterebbe nel mondo reale. Durante lo sviluppo del modello, il set di convalida dovrebbe essere utilizzato per tutte le attività di ottimizzazione dei parametri, selezione del modello, arresto anticipato e simili. È importante riservare una parte, come il 20-30% dell'intero set di dati, esclusivamente per la valutazione finale del modello. Questi dati non dovrebbero essere utilizzati per la convalida, l'ottimizzazione o la selezione del modello durante lo sviluppo.

Non mantenere un set "hold-out" non visto per la convalida finale rischia di ottimizzare i risultati e trascurare potenziali errori prima del rilascio del modello. Avere alcuni dati nuovi fornisce un controllo di integrità finale sull'efficacia nel mondo reale. Mantenere la completa separazione di addestramento/validazione dal set di test è essenziale per ottenere stime accurate delle prestazioni del modello. Anche piccole deviazioni da un singolo utilizzo del set di test potrebbero falsare positivamente i risultati e le metriche, fornendo una visione eccessivamente ottimistica dell'efficacia nel mondo reale.

7.4.2.5 Stesse Suddivisioni dei Dati negli Esperimenti

Quando si confrontano diversi modelli di machine learning o si sperimentano varie architetture e iperparametri, utilizzare le stesse suddivisioni dei dati per l'addestramento, la validazione e il test nei diversi esperimenti può introdurre distorsioni e invalidare le comparazioni.

Se le stesse suddivisioni vengono riutilizzate, i risultati della valutazione potrebbero essere più bilanciati e misurare accuratamente quale modello funziona meglio. Ad esempio, una certa suddivisione casuale dei dati potrebbe favorire il modello A rispetto al modello B indipendentemente dagli algoritmi. Riutilizzare questa suddivisione causerà quindi distorsioni a favore del modello A.

Invece, le suddivisioni dei dati dovrebbero essere randomizzate o mescolate per ogni iterazione sperimentale. Ciò garantisce che la casualità nel campionamento delle suddivisioni non conferisca un vantaggio ingiusto a nessun modello.

Con diverse suddivisioni per esperimento, la valutazione diventa più solida. Ogni modello viene testato su un'ampia gamma di set di test estratti casualmente dalla popolazione complessiva, attenuando la variazione e rimuovendo la correlazione tra i risultati.

La prassi corretta è quella di impostare un “seed” casuale prima di suddividere i dati per ogni esperimento. La suddivisione dovrebbe avvenire dopo il rimescolamento/ricampionamento come parte della pipeline sperimentale. Eseguire confronti sulle stesse suddivisioni viola l’ipotesi i.i.d (indipendenti e identicamente distribuite) richiesta per la validità statistica.
Le suddivisioni univoche sono essenziali per confronti di modelli equi. Sebbene richieda un’elaborazione più intensiva, l’allocazione randomizzata per esperimento rimuove la distorsione del campionamento e consente un benchmarking valido. Ciò evidenzia le vere differenze nelle prestazioni del modello indipendentemente dalle caratteristiche di una particolare suddivisione.

7.4.2.6 Mancata Stratificazione delle Suddivisioni

Quando si suddividono i dati in set di training, validazione e test, la mancata stratificazione delle suddivisioni può comportare una rappresentazione non uniforme delle classi target tra le suddivisioni e introdurre un bias di campionamento. Ciò è particolarmente problematico per i set di dati sbilanciati.
La suddivisione stratificata implica il campionamento dei dati in modo che la proporzione di classi di output sia approssimativamente preservata in ogni suddivisione. Ad esempio, se si esegue una suddivisione training-test 70/30 su un set di dati con campioni negativi al 60% e positivi al 40%, la stratificazione garantisce esempi negativi al ~60% e positivi al ~40% sia nei set di training che nei set di test.
Senza stratificazione, la casualità potrebbe comportare che la suddivisione di training abbia campioni positivi al 70% mentre il test ha campioni positivi al 30%. Il modello addestrato su questa distribuzione di training distorta non si generalizzerà bene. Lo squilibrio delle classi compromette anche le metriche del modello come l’accuratezza.
La stratificazione funziona meglio quando viene eseguita utilizzando etichette, sebbene proxy come il clustering possano essere utilizzati per l’apprendimento non supervisionato. Diventa essenziale per set di dati altamente distorti con classi rare che potrebbero essere facilmente omesse dalle suddivisioni.
Librerie come Scikit-Learn hanno metodi di suddivisione stratificati nativi. Non utilizzarli potrebbe inavvertitamente introdurre bias di campionamento e danneggiare le prestazioni del modello sui gruppi minoritari. Dopo aver eseguito le suddivisioni, il bilanciamento complessivo delle classi dovrebbe essere esaminato per garantire una rappresentazione uniforme tra le suddivisioni.
La stratificazione fornisce un set di dati bilanciato sia per l’addestramento del modello che per la valutazione. Sebbene la semplice suddivisione casuale sia facile, tenendo conto delle esigenze di stratificazione, specialmente per dati sbilanciati nel mondo reale, si traduce in uno sviluppo e una valutazione del modello più solidi.

7.4.2.7 Ignorare le Dipendenze delle Serie Temporali

I dati delle serie temporali hanno una struttura temporale intrinseca con osservazioni dipendenti dal contesto passato. Suddividere ingenuamente i dati delle serie temporali in set di training e test senza tenere conto di questa dipendenza porta a perdite di dati e bias di lookahead.
Ad esempio, suddividere semplicemente una serie temporale nel primo 70% di training e nell’ultimo 30% come dati di test contaminerà i dati di training con dati futuri. Il modello può usare queste informazioni per “sbirciare” in avanti durante il training.
Ciò si traduce in una valutazione eccessivamente ottimistica delle prestazioni del modello. Il modello può sembrare che preveda il futuro in modo accurato, ma in realtà ha appreso implicitamente in base ai dati futuri, il che non si traduce in prestazioni nel mondo reale.
Dovrebbero essere utilizzate tecniche di validazione incrociata delle serie temporali appropriate, come il concatenamento in avanti, per preservare l’ordine e la dipendenza. Il set

di test dovrebbe contenere solo dati da una finestra temporale futura a cui il modello non è stato esposto per il training.
Non tenere conto delle relazioni temporali porta a ipotesi di causalità non valide. Se i dati di training contengono dati futuri, il modello potrebbe anche dover imparare come estrapolare ulteriormente le previsioni.
Mantenere il flusso temporale degli eventi ed evitare il bias di lookahead è fondamentale per addestrare e testare correttamente i modelli di serie temporali. Ciò garantisce che possano davvero prevedere pattern futuri e non solo memorizzare i dati di training passati.

7.4.2.8 Nessun Dato Non Visto per la Valutazione Finale

Un errore comune quando si suddividono i dati è non metterne da parte una porzione solo per la valutazione finale del modello completato. Tutti i dati vengono utilizzati per training, validazione e set di test durante lo sviluppo.
Questo non lascia dati non visti per ottenere una stima imparziale di come il modello finale ottimizzato si comporterebbe nel mondo reale. Le metriche sul set di test utilizzate durante lo sviluppo potrebbero riflettere solo parzialmente le reali capacità del modello.
Ad esempio, scelte come l'arresto anticipato e l'ottimizzazione degli iperparametri sono spesso ottimizzate in base alle prestazioni del set di test. Questo accoppia il modello ai dati di test. È necessario un set di dati non visto per interrompere questo accoppiamento e ottenere metriche reali del mondo reale.
La "best practice" è quella di riservare una parte, come il 20-30% del set di dati completo, esclusivamente per la valutazione finale del modello. Questi dati non dovrebbero essere utilizzati per la convalida, l'ottimizzazione o la selezione del modello durante lo sviluppo.
Il salvataggio di alcuni dati non visti consente di valutare il modello completamente addestrato come una scatola nera su dati del mondo reale. Questo fornisce metriche affidabili per decidere se il modello è pronto per la distribuzione in produzione.
Non mantenere un set "hold-out" non visto per la convalida finale rischia di ottimizzare i risultati e trascurare potenziali errori prima del rilascio del modello. Avere alcuni dati nuovi fornisce un controllo di integrità finale sull'efficacia nel mondo reale.

7.4.2.9 Sovra-ottimizzazione del Set di Validazione

Il set di validazione è pensato per guidare il processo di training del modello, non per fungere da dati di training aggiuntivi. L'eccessiva ottimizzazione del set di validazione per massimizzare le metriche delle prestazioni lo tratta più come un set di training secondario, portando a metriche gonfiate e scarsa generalizzazione.
Ad esempio, tecniche come l'ottimizzazione estensiva degli iperparametri o l'aggiunta di incrementi di dati mirati a migliorare l'accuratezza della convalida possono far sì che il modello si adatti troppo ai dati di validazione. Il modello può raggiungere un'accuratezza di validazione del 99% ma solo un'accuratezza di test del 55%.
Analogamente, riutilizzare il set di validazione per un arresto anticipato può anche ottimizzare il modello specificamente per quei dati. L'arresto alle migliori prestazioni di validazione sovra-adatta il rumore e le fluttuazioni causate dalle piccole dimensioni di validazione.
Il set di validazione funge da proxy per ottimizzare e selezionare i modelli. Tuttavia, l'obiettivo rimane massimizzare le prestazioni dei dati del mondo reale, non il set di validazione. Ridurre al minimo la perdita o l'errore sui dati di validazione non si traduce automaticamente in una buona generalizzazione.
Un buon approccio è quello di mantenere l'uso del set di validazione al minimo: gli iperparametri possono essere regolati grossolanamente prima sui dati di training, ad esempio. Il set di validazione guida il training ma non dovrebbe influenzare o alterare il modello stesso. È uno strumento diagnostico, non di ottimizzazione.

Quando si valutano le prestazioni sul set di validazione, bisogna fare attenzione a non sovra-adattare. Sono necessari dei compromessi per costruire modelli che funzionino bene sulla popolazione complessiva e non siano eccessivamente regolati sui campioni di validazione.

7.5 Algoritmi di Ottimizzazione

Stochastic gradient descent (SGD) è un algoritmo di ottimizzazione semplice ma potente per l'addestramento di modelli di machine learning. Funziona stimando il gradiente della funzione di perdita relativa ai parametri del modello utilizzando un singolo esempio di addestramento e poi aggiornando i parametri nella direzione che riduce la perdita.
Sebbene concettualmente semplice, SGD necessita di alcune aree di miglioramento. Innanzitutto, scegliere un tasso di apprendimento appropriato può essere difficile: troppo piccolo e i progressi sono molto lenti; troppo grande e i parametri possono oscillare e non convergere. In secondo luogo, SGD tratta tutti i parametri in modo uguale e indipendente, il che potrebbe non essere l'ideale in tutti i casi. Infine, SGD vanilla [standard] utilizza solo informazioni sul gradiente di primo ordine, il che si traduce in progressi lenti su problemi mal condizionati.

7.5.1 Ottimizzazioni

Nel corso degli anni, sono state proposte varie ottimizzazioni per accelerare e migliorare l'SGD vanilla. Ruder (2016) fornisce un'eccellente panoramica dei diversi ottimizzatori. In breve, diverse tecniche di ottimizzazione SGD comunemente utilizzate includono:

Momentum: Accumula un vettore di velocità in direzioni di gradiente persistente attraverso le iterazioni. Ciò aiuta ad accelerare i progressi smorzando le oscillazioni e mantiene i progressi in direzioni coerenti.

Nesterov Accelerated Gradient (NAG): Una variante di momentum che calcola i gradienti in "look ahead" anziché nella posizione del parametro corrente. Questo aggiornamento anticipatorio impedisce l'overshooting mentre il momentum mantiene il progresso accelerato.

Adagrad: Un algoritmo di velocità di apprendimento adattivo che mantiene una velocità di apprendimento per parametro ridotta proporzionalmente alla somma storica dei gradienti di ciascun parametro. Aiuta a eliminare la necessità di regolare manualmente i tassi di apprendimento (Duchi, Hazan, e Singer 2010).

Adadelta: Una modifica ad Adagrad limita la finestra dei gradienti passati accumulati, riducendo così il decadimento aggressivo dei tassi di apprendimento (Zeiler 2012).

RMSProp: Divide il tasso di apprendimento per una media esponenzialmente decrescente dei gradienti quadrati. Ciò ha un effetto di normalizzazione simile ad Adagrad ma non accumula i gradienti nel tempo, evitando un rapido decadimento dei tassi di apprendimento (Hinton 2017).

Adam: Combinazione di momentum e rmsprop dove rmsprop modifica il tasso di apprendimento in base alla media delle recenti ampiezze dei gradienti. Mostra un progresso iniziale molto rapido e regola automaticamente le dimensioni dei passi (Kingma e Ba 2014).

AMSGrad: Una variante di Adam che assicura una convergenza stabile mantenendo il massimo dei gradienti quadratici passati, impedendo al tasso di apprendimento di aumentare durante l'addestramento (S. J. Reddi, Kale, e Kumar 2019).
Tra questi metodi, Adam è ampiamente considerato l'algoritmo di ottimizzazione di riferimento per molte attività di deep-learning. Supera costantemente SGD vanilla in termini

di velocità di addestramento e prestazioni. Altri ottimizzatori potrebbero essere più adatti in alcuni casi, in particolare per modelli più semplici.

7.5.2 Compromessi

Tabella 7.2 è una tabella di pro e contro per alcuni dei principali algoritmi di ottimizzazione per l'addestramento di reti neurali:

Tabella 7.2: Confronto dei pro e dei contro di diversi algoritmi di ottimizzazione.

Algoritmo	Pro	Contro
Momentum	• Convergenza più rapida dovuta all'accelerazione lungo i gradienti • Minore oscillazione rispetto a SGD vanilla	• Richiede la messa a punto del parametro momentum
Nesterov Accelerated Gradient (NAG)	• Più veloce dello slancio standard in alcuni casi • Gli aggiornamenti anticipati impediscono il superamento	• Più complesso da comprendere intuitivamente
Adagrad	• Elimina la necessità di regolare manualmente i tassi di apprendimento • Funziona bene su gradienti radi	• Il tasso di apprendimento potrebbe decadere troppo rapidamente su gradienti densi
Adadelta	• Decadimento del tasso di apprendimento meno aggressivo rispetto ad Adagrad	• Ancora sensibile al valore iniziale del tasso di apprendimento
RMSProp	• Regola automaticamente i tassi di apprendimento • Funziona bene nella pratica	• Nessun aspetto negativo importante
Adam	• Combinazione di momentum e tassi di apprendimento adattivo • Convergenza efficiente e veloce	• Prestazioni di generalizzazione leggermente peggiori in alcuni casi
AMSGrad	• Miglioramento di Adam che affronta il problema della generalizzazione	• Non è stato utilizzato/testato così ampiamente come Adam

7.5.3 Algoritmi di Benchmarking

Non esiste un singolo metodo migliore per tutti i tipi di problemi. Ciò significa che abbiamo bisogno di un benchmarking completo per identificare l'ottimizzatore più efficace per set di dati e modelli specifici. Le prestazioni di algoritmi come Adam, RMSProp e Momentum variano a seconda delle dimensioni del batch, dei programmi di apprendimento, dell'architettura del modello, della distribuzione dei dati e della regolarizzazione. Queste variazioni sottolineano l'importanza di valutare ogni ottimizzatore in diverse condizioni.

Prendiamo ad esempio Adam, che spesso eccelle nelle attività di visione artificiale, a differenza di RMSProp, che potrebbe mostrare una migliore generalizzazione in determinate attività di elaborazione del linguaggio naturale. La forza di Momentum risiede nella sua accelerazione in scenari con direzioni di gradiente coerenti, mentre i tassi di apprendimento adattivo di Adagrad sono più adatti per problemi di gradiente sparso.
Questa vasta gamma di interazioni tra ottimizzatori dimostra la difficoltà di dichiarare un singolo algoritmo universalmente superiore. Ogni ottimizzatore ha punti di forza unici, rendendo fondamentale valutare vari metodi per scoprire empiricamente le loro condizioni di applicazione ottimali.
Un approccio di benchmarking completo dovrebbe valutare la velocità di convergenza e fattori come errore di generalizzazione, stabilità, sensibilità degli iperparametri ed efficienza computazionale, tra gli altri. Ciò comporta il monitoraggio delle curve di apprendimento di training e convalida su più esecuzioni e il confronto degli ottimizzatori su vari set di dati e modelli per comprenderne i punti di forza e di debolezza.
AlgoPerf, introdotto da Dürr et al. (2021), risponde alla necessità di un sistema di benchmarking robusto. Questa piattaforma valuta le prestazioni dell'ottimizzatore utilizzando criteri quali curve di loss [perdita] di training, errore di generalizzazione, sensibilità agli iperparametri ed efficienza computazionale. AlgoPerf testa vari metodi di ottimizzazione, tra cui Adam, LAMB e Adafactor, su diversi tipi di modelli come CNN e RNN/LSTM su set di dati stabiliti. Utilizza la "containerizzazione" e la raccolta automatica di metriche per ridurre al minimo le incongruenze e consente esperimenti controllati su migliaia di configurazioni, fornendo una base affidabile per confrontare gli ottimizzatori.
Le informazioni ottenute da AlgoPerf e benchmark simili sono inestimabili per guidare la scelta ottimale o la messa a punto degli ottimizzatori. Abilitando valutazioni riproducibili, questi benchmark contribuiscono a una comprensione più approfondita delle prestazioni di ciascun ottimizzatore, aprendo la strada a innovazioni future e progressi accelerati nel settore.

7.6 Ottimizzazione degli Iperparametri

Gli iperparametri sono impostazioni importanti nei modelli di machine learning che incidono notevolmente sulle prestazioni finali dei modelli. A differenza di altri parametri del modello che vengono appresi durante l'addestramento, gli iperparametri vengono specificati dai "data scientist" o dagli ingegneri del machine learning prima dell'addestramento del modello.
La scelta dei valori degli iperparametri corretti consente ai modelli di apprendere pattern dai dati in modo efficace. Alcuni esempi di iperparametri chiave negli algoritmi di apprendimento automatico includono:

- **Reti neurali:** Velocità di apprendimento, dimensione del batch, numero di unità nascoste, funzioni di attivazione
- **Macchine a vettori di supporto:** Forza di regolarizzazione, tipo di kernel e parametri
- **Random forest:** Numero di alberi, profondità dell'albero
- **K-means:** Numero di cluster

Il problema è che non ci sono regole pratiche affidabili per scegliere configurazioni ottimali degli iperparametri: in genere si devono provare valori diversi e valutare le prestazioni. Questo processo è chiamato "hyperparameter tuning" ottimizzazione degli iperparametri.
Nei primi anni del moderno deep learning, i ricercatori erano ancora alle prese con problemi di convergenza instabile e lenta. I punti dolenti comuni includevano perdite di training che fluttuavano selvaggiamente, gradienti che esplodevano o svanivano e un'ampia serie di tentativi ed errori necessari per addestrare le reti in modo affidabile. Di conseguenza, un

punto focale iniziale era l'utilizzo di iperparametri per controllare l'ottimizzazione del modello. Ad esempio, tecniche seminali come la normalizzazione batch consentivano una convergenza più rapida del modello regolando gli aspetti dello spostamento interno delle covariate. I metodi di velocità di apprendimento adattivo hanno anche mitigato la necessità di estese pianificazioni manuali. Questi affrontavano problemi di ottimizzazione durante l'addestramento, come la divergenza incontrollata del gradiente. Le velocità di apprendimento adattate con attenzione sono anche il fattore di controllo primario per ottenere una convergenza rapida e stabile anche oggi.
Con l'espansione esponenziale della capacità computazionale negli anni successivi, modelli molto più grandi potevano essere addestrati senza cadere preda di problemi di pura ottimizzazione numerica. L'attenzione si è spostata verso la generalizzazione, sebbene una convergenza efficiente fosse un prerequisito fondamentale. Tecniche all'avanguardia come "Transformers" hanno introdotto miliardi di parametri. A tali dimensioni, gli iperparametri relativi a capacità, regolarizzazione, ensembling [raggruppamento], ecc., hanno assunto un ruolo centrale per la messa a punto, anziché solo le metriche di convergenza grezze.
La lezione è che comprendere l'accelerazione e la stabilità del processo di ottimizzazione stesso costituisce il lavoro di base. Schemi di inizializzazione, dimensioni dei batch, decadimenti di peso e altri iperparametri di training rimangono indispensabili oggi. Dominare una convergenza rapida e impeccabile consente ai professionisti di espandere la propria attenzione sulle esigenze emergenti relative alla messa a punto di parametri quali accuratezza, robustezza ed efficienza su larga scala.

7.6.1 Algoritmi di Ricerca

Quando si tratta del processo critico di ottimizzazione degli iperparametri, ci sono diversi algoritmi sofisticati su cui gli specialisti del machine learning si affidano per cercare sistematicamente nel vasto spazio di possibili configurazioni dei modelli. Alcuni degli algoritmi di ricerca degli iperparametri più importanti includono:

- **Grid Search:** Il metodo di ricerca più elementare, in cui si definisce manualmente una griglia di valori da controllare per ogni iperparametro. Ad esempio, controllando `velocità di apprendimento = [0.01, 0.1, 1]` e `dimensioni batch = [32, 64, 128]`. Il vantaggio principale è la semplicità, ma può portare a un'esplosione esponenziale nello spazio di ricerca, rendendolo dispendioso in termini di tempo. È più adatto per l'ottimizzazione di un piccolo numero di parametri.
- **Random Search:** Invece di definire una griglia, si selezionano casualmente valori per ogni iperparametro da un intervallo o set predefinito. Questo metodo è più efficiente nell'esplorazione di un vasto spazio di iperparametri perché non richiede una ricerca esaustiva. Tuttavia, potrebbe comunque non trovare parametri ottimali poiché non esplora sistematicamente tutte le possibili combinazioni.
- **Bayesian Optimization:** Questo è un approccio probabilistico avanzato per l'esplorazione adattiva basato su una funzione surrogata per modellare le prestazioni su iterazioni. È semplice ed efficiente: trova iperparametri altamente ottimizzati in meno passaggi di valutazione. Tuttavia, richiede un maggiore investimento nella configurazione (Snoek, Larochelle, e Adams 2012).
- **Evolutionary Algorithms:** Questi algoritmi imitano i principi della selezione naturale. Generano popolazioni di combinazioni di iperparametri e le evolvono nel tempo in base alle prestazioni. Questi algoritmi offrono solide capacità di ricerca più adatte per superfici di risposta complesse. Tuttavia, sono necessarie molte iterazioni per una convergenza ragionevole.
- **Population Based Training (PBT):** Un metodo che ottimizza gli iperparametri addestrando più modelli in parallelo, consentendo loro di condividere e adattare

configurazioni di successo durante l'addestramento, combinando elementi di ricerca casuale e algoritmi evolutivi (Jaderberg et al. 2017).

- **Neural Architecture Search:** Un approccio alla progettazione di architetture ad alte prestazioni per reti neurali. Tradizionalmente, gli approcci NAS utilizzano una qualche forma di apprendimento di rinforzo per proporre architetture di reti neurali, che vengono poi ripetutamente valutate (Zoph e Le 2016).

7.6.2 Implicazioni di Sistema

La messa a punto degli iperparametri può avere un impatto significativo sul tempo di convergenza durante l'addestramento del modello, influenzando direttamente il runtime complessivo. I valori corretti per gli iperparametri chiave di training sono cruciali per un'efficiente convergenza del modello. Ad esempio, la velocità di apprendimento dell'iperparametro controlla la dimensione del passo durante l'ottimizzazione della discesa del gradiente. Impostando correttamente uno scheduling della velocità di apprendimento assicura che l'algoritmo di ottimizzazione converga rapidamente verso un buon minimo. Una velocità di apprendimento troppo bassa porta a una convergenza dolorosamente lenta, mentre un valore troppo grande causa una fluttuazione selvaggia delle perdite. Una messa a punto corretta assicura un rapido movimento verso pesi e bias ottimali.
Analogamente, la dimensione del batch per la discesa del gradiente stocastica influisce sulla stabilità della convergenza. La giusta dimensione del batch attenua le fluttuazioni negli aggiornamenti dei parametri per avvicinarsi più rapidamente al minimo. Sono necessarie più dimensioni del batch per evitare una convergenza rumorosa, mentre le dimensioni maggiori del batch non riescono a generalizzare e rallentano la convergenza a causa di aggiornamenti dei parametri meno frequenti. La messa a punto degli iperparametri per una convergenza più rapida e una durata di addestramento ridotta ha implicazioni dirette sui costi e sui requisiti di risorse per il ridimensionamento dei sistemi di machine learning:

- **Costi computazionali inferiori:** Tempi di convergenza più brevi significano costi computazionali inferiori per i modelli di training. Il training ML sfrutta spesso grandi istanze di cloud computing come cluster GPU e TPU che comportano pesanti costi orari. Ridurre al minimo i tempi di training riduce direttamente questo costo di noleggio delle risorse, che tende a dominare i budget ML per le organizzazioni. Un'iterazione più rapida consente inoltre agli esperti di dati di sperimentare più liberamente all'interno dello stesso budget.
- **Tempo di training ridotto:** Un tempo di training ridotto sblocca opportunità per addestrare più modelli utilizzando lo stesso budget computazionale. Gli iperparametri ottimizzati estendono ulteriormente le risorse disponibili, consentendo alle aziende di sviluppare e sperimentare più modelli con vincoli di risorse per massimizzare le prestazioni.
- **Efficienza delle risorse:** Un training più rapido consente di allocare istanze di calcolo più piccole nel cloud poiché i modelli richiedono l'accesso alle risorse per una durata più breve. Ad esempio, un job di training di un'ora consente di utilizzare istanze GPU meno potenti rispetto a un training di più ore, che richiede un accesso di elaborazione sostenuto su intervalli più lunghi. Ciò consente di risparmiare sui costi, soprattutto per carichi di lavoro di grandi dimensioni.

Ci sono anche altri vantaggi. Ad esempio, una convergenza più rapida riduce la pressione sui team di ingegneria ML in merito al provisioning delle risorse di training. Le semplici routine di riaddestramento del modello possono utilizzare risorse meno potenti anziché richiedere l'accesso a code ad alta priorità per cluster GPU di livello di produzione vincolati, liberando risorse di distribuzione per altre applicazioni.

7.6.3 Gli Auto Tuner

Data la sua importanza, esiste un'ampia gamma di offerte commerciali per aiutare con l'ottimizzazione degli iperparametri. Toccheremo brevemente due esempi: uno incentrato sull'ottimizzazione per ML su scala cloud e l'altro per modelli di apprendimento automatico mirati ai microcontrollori. Tabella 7.3 delinea le principali differenze:

Tabella 7.3: Confronto di piattaforme di ottimizzazione per diversi casi d'uso di machine learning.

Piattaforma	Caso d'Uso Target	Tecniche di ottimizzazione	Vantaggi
Vertex AI di Google	Apprendimento automatico su scala cloud	Ottimizzazione bayesiana, addestramento Population-Based	Nasconde la complessità, consentendo modelli rapidi e pronti per l'implementazione con ottimizzazione iperparametrica all'avanguardia
EON Tuner di Edge Impulse	Modelli di microcontrollori (TinyML)	Ottimizzazione bayesiana	Adatta i modelli per dispositivi con risorse limitate, semplifica l'ottimizzazione per l'implementazione embedded

7.6.3.1 BigML

Sono disponibili diverse piattaforme commerciali di auto-tuning per risolvere questo problema. Una soluzione è Vertex AI Cloud di Google, che offre un ampio supporto integrato per tecniche di ottimizzazione all'avanguardia.

Una delle funzionalità più importanti della piattaforma di apprendimento automatico gestita da Vertex AI di Google è l'ottimizzazione efficiente e integrata degli iperparametri per lo sviluppo del modello. Per addestrare con successo modelli ML performanti è necessario identificare configurazioni ottimali per un set di iperparametri esterni che determinano il comportamento del modello, ponendo un problema di ricerca ad alta dimensione impegnativo. Vertex AI semplifica questo processo tramite strumenti di Automated Machine Learning (AutoML).

In particolare, gli scienziati dei dati possono sfruttare i motori di ottimizzazione degli iperparametri di Vertex AI fornendo un set di dati etichettato e scegliendo un tipo di modello come un classificatore di reti neurali o Random Forest. Vertex avvia un job di "Hyperparameter Search" in modo trasparente sul backend, gestendo completamente il provisioning delle risorse, l'addestramento del modello, il monitoraggio delle metriche e l'analisi dei risultati automaticamente utilizzando algoritmi di ottimizzazione avanzati.

Internamente, Vertex AutoML impiega varie strategie di ricerca per esplorare in modo intelligente le configurazioni di iperparametri più promettenti in base ai risultati delle valutazioni precedenti. Tra queste, l'ottimizzazione bayesiana è offerta in quanto fornisce un'efficienza di campionamento superiore, richiedendo meno iterazioni di training per ottenere una qualità del modello ottimizzata rispetto ai metodi standard di Grid Search o di Random Search. Per spazi di ricerca di architettura neurale più complessi, Vertex AutoML utilizza il Population-Based Training, che addestra simultaneamente più modelli e regola dinamicamente i loro iperparametri sfruttando le prestazioni di altri modelli nella popolazione, analogamente ai principi di selezione naturale.

Vertex AI democratizza le tecniche di ricerca di iperparametri all'avanguardia su scala cloud per tutti gli sviluppatori ML, astraendo la complessità di esecuzione e di orchestrazione sottostante. Gli utenti si concentrano esclusivamente sul loro set di dati, sui requisiti del modello e sugli obiettivi di accuratezza, mentre Vertex gestisce il ciclo di ottimizzazione, l'allocazione delle risorse, il training del modello, il monitoraggio dell'accuratezza e l'archiviazione degli artefatti internamente. Il risultato è ottenere modelli ML ottimizzati e pronti per la distribuzione più velocemente per il problema target.

7.6.3.2 TinyML

Edge Impulse's Efficient On-device Neural Network Tuner (EON Tuner) è uno strumento di ottimizzazione automatizzata degli iperparametri progettato per sviluppare modelli di apprendimento automatico per microcontrollori. Semplifica il processo di sviluppo del modello trovando automaticamente la migliore configurazione di rete neurale per un'implementazione efficiente e accurata su dispositivi con risorse limitate.

La funzionalità chiave di EON Tuner è la seguente. Innanzitutto, gli sviluppatori definiscono gli iperparametri del modello, come numero di layer, nodi per layer, funzioni di attivazione e pianificazione della velocità di "annealing" [https://it.wikipedia.org/wiki/Ricottura_simulata] dell'apprendimento. Questi parametri costituiscono lo spazio di ricerca che verrà ottimizzato. Successivamente, viene selezionata la piattaforma del microcontrollore target, fornendo vincoli hardware embedded. L'utente può anche specificare obiettivi di ottimizzazione, come la riduzione dell'ingombro di memoria, la riduzione della latenza, la riduzione del consumo energetico o la massimizzazione della precisione.

Con lo spazio di ricerca definito e gli obiettivi di ottimizzazione, EON Tuner sfrutta l'ottimizzazione degli iperparametri bayesiani per esplorare in modo intelligente possibili configurazioni. Ogni configurazione potenziale viene automaticamente implementata come specifica di modello completa, addestrata e valutata per metriche di qualità. Il processo continuo bilancia esplorazione e sfruttamento per arrivare a impostazioni ottimizzate su misura per l'architettura del chip scelta dallo sviluppatore e i requisiti di prestazioni.

EON Tuner libera gli esperti di machine learning dal processo iterativo esigente di messa a punto manuale dei modelli, regolando automaticamente i modelli per il deployment embedded. Lo strumento si integra perfettamente nel flusso di lavoro Edge Impulse, portando i modelli dal concetto a implementazioni ottimizzate in modo efficiente sui microcontrollori. L'esperienza racchiusa in EON Tuner per quanto riguarda l'ottimizzazione del modello ML per i microcontrollori garantisce che sia gli sviluppatori principianti che quelli esperti possano rapidamente iterare per ottenere modelli adatti alle esigenze del loro progetto.

Esercizio 7.2: Ottimizzazione degli Iperparametri

Prepariamoci a scoprire i segreti della messa a punto degli iperparametri e portiamo i modelli PyTorch al livello successivo! Gli iperparametri sono come i quadranti e le manopole nascosti che controllano i superpoteri di apprendimento del modello. In questo notebook Colab, si collaborerà con Ray Tune per trovare le combinazioni perfette di iperparametri. Scopriamo come definire quali valori cercare, impostare il codice di training per l'ottimizzazione e lasciare che Ray Tune faccia il grosso del lavoro. Alla fine, si diventerà professionisti della messa a punto degli iperparametri!

Open in Colab

Video 7.3 spiega l'organizzazione sistematica del processo di ottimizzazione degli iperparametri.

Video 7.3: Iperparametro

https://www.youtube.com/watch?v=AXDByU3D1hA&list=PLkDaE6sCZn6Hn0vK8co82zjQtt3T2Nkqc&

7.7 Regolarizzazione

La regolarizzazione è una tecnica critica per migliorare le prestazioni e la generalizzabilità dei modelli di machine learning in impostazioni applicate. Si riferisce alla limitazione matematica o alla penalizzazione della complessità del modello per evitare il sovra-adattamento dei dati di training. Senza regolarizzazione, i modelli ML complessi sono inclini al sovra-adattamento del set di dati e alla memorizzazione di peculiarità e rumore nel set di training anziché all'apprendimento di pattern significativi. Possono raggiungere un'elevata accuratezza di training ma hanno prestazioni scadenti quando valutano nuovi input non ancora visti.
La regolarizzazione aiuta ad affrontare questo problema ponendo vincoli che favoriscono modelli più semplici e più generalizzabili che non si agganciano a errori di campionamento. Tecniche come la regolarizzazione L1/L2 penalizzano direttamente valori di parametri elevati durante il training, costringendo il modello a utilizzare i parametri più piccoli che possono spiegare adeguatamente il segnale. Le regole di arresto anticipato interrompono il training quando le prestazioni del set di validazione smettono di migliorare, prima che il modello inizi a sovra-adattarsi.
Una regolarizzazione appropriata è fondamentale quando si distribuiscono modelli a nuove popolazioni di utenti e ambienti in cui sono probabili cambiamenti di distribuzione. Ad esempio, un modello irregolare di rilevamento delle frodi addestrato presso una banca potrebbe funzionare inizialmente, ma accumulare debiti tecnici nel tempo man mano che emergono nuovi pattern di frode.
La regolarizzazione di reti neurali complesse offre anche vantaggi computazionali: modelli più piccoli richiedono meno "data augmentation", potenza di calcolo e archiviazione dei dati. La regolarizzazione consente anche sistemi di intelligenza artificiale più efficienti, in cui accuratezza, robustezza e gestione delle risorse sono attentamente bilanciate rispetto alle limitazioni del set di addestramento.
Diverse potenti tecniche di regolarizzazione sono comunemente utilizzate per migliorare la generalizzazione del modello. L'architettura della strategia ottimale richiede la comprensione di come ogni metodo influisce sull'apprendimento e sulla complessità del modello.

7.7.1 L1 e L2

Due delle forme di regolarizzazione più ampiamente utilizzate sono la regolarizzazione L1 e la L2. Entrambe penalizzano la complessità del modello aggiungendo un termine extra alla funzione di costo ottimizzata durante l'addestramento. Questo termine cresce all'aumentare dei parametri del modello.
La regolarizzazione L2, nota anche come "ridge regression" [https://it.wikipedia.org/wiki/Regolarizzazione_di_Tichonov], aggiunge la somma delle grandezze al quadrato di tutti i parametri moltiplicata per un coefficiente α. Questa penalità quadratica riduce i valori dei parametri estremi in modo più aggressivo rispetto alle tecniche L1. L'implementazione richiede solo la modifica della funzione di costo e la messa a punto di α.

$$R_{L2}(\Theta) = \alpha \sum_{i=1}^{n} \theta_i^2$$

Dove:

- $R_{L2}(\Theta)$ - Il termine di regolarizzazione L2 che viene aggiunto alla funzione di costo
- α - L'iperparametro di regolarizzazione L2 che controlla la forza della regolarizzazione
- θ_i - L'i-esimo parametro del modello

- n - Il numero di parametri nel modello
- θ_i^2 - Il quadrato di ciascun parametro

E la funzione di costo regolarizzata L2 completa è:

$$J(\theta) = L(\theta) + R_{L2}(\Theta)$$

Dove:

- $L(\theta)$ - La funzione di costo non regolarizzata originale
- $J(\theta)$ - La nuova funzione di costo regolarizzata

Sia la regolarizzazione L1 che L2 penalizzano i pesi elevati nella rete neurale. Tuttavia, la differenza fondamentale tra la regolarizzazione L1 e L2 è che la regolarizzazione L2 penalizza i quadrati dei parametri anziché i valori assoluti. Questa differenza fondamentale ha un impatto considerevole sui pesi regolarizzati risultanti. La regolarizzazione L1, o regressione LASSO [https://it.wikipedia.org/wiki/Regolarizzazione_(matematica)], utilizza la somma assoluta delle grandezze anziché il quadrato moltiplicato per α. La penalizzazione del valore assoluto dei pesi induce scarsità poiché il gradiente degli errori estrapola linearmente mentre i termini dei pesi tendono a zero; questo è diverso dalla penalizzazione del valore al quadrato dei pesi, dove la penalità si riduce man mano che i pesi tendono a 0. Inducendo scarsità nel vettore dei parametri, la regolarizzazione L1 esegue automaticamente la selezione delle feature, impostando i pesi delle feature irrilevanti a zero. A differenza della regolarizzazione L2, la L1 porta alla scarsità poiché i pesi sono impostati su 0; nella regolarizzazione L2, i pesi sono impostati su un valore molto vicino a 0 ma generalmente non raggiungono mai esattamente 0. La regolarizzazione L1 incoraggia la scarsità ed è stata utilizzata in alcuni lavori per addestrare reti sparse che potrebbero essere più efficienti in termini di hardware (Hoefler et al. 2021).

$$R_{L1}(\Theta) = \alpha \sum_{i=1}^{n} | \quad |\theta_i| \quad |$$

Dove:

- $R_{L1}(\Theta)$ - Il termine di regolarizzazione L1 che viene aggiunto alla funzione di costo
- α - L'iperparametro di regolarizzazione L1 che controlla la forza della regolarizzazione
- θ_i - L'i-esimo parametro del modello
- n - Il numero di parametri nel modello
- $| \quad |\theta_i| \quad |$ - La norma L1, che assume il valore assoluto di ciascun parametro

E la funzione di costo regolarizzata L1 completa è:

$$J(\theta) = L(\theta) + R_{L1}(\Theta)$$

Dove:

- $L(\theta)$ - La funzione di costo non regolarizzata originale
- $J(\theta)$ - La nuova funzione di costo regolarizzata

La scelta tra L1 e L2 dipende dalla complessità del modello prevista e dalla necessità o meno di una selezione di feature intrinseche. Entrambi richiedono una messa a punto iterativa su un set di validazione per selezionare l'iperparametro α ottimale.

Video 7.4 e Video 7.5 spiegano come funziona la regolarizzazione.

Video 7.4: Regolarizzazione

https://www.youtube.com/watch?v=6g0t3Phly2M&list=PLkDaE6sCZn6Hn0vK8co82zjQtt3T2Nkqc&index=4

Video 7.5 spiega come la regolarizzazione può aiutare a ridurre l'overfitting del modello per migliorare le prestazioni.

Video 7.5: Perché la Regolarizzazione Riduce l'Overfitting

https://www.youtube.com/watch?v=NyG-7nRpsW8&list=PLkDaE6sCZn6Hn0vK8co82zjQtt3T2Nkqc&index=5

7.7.2 Dropout

Un altro metodo di regolarizzazione ampiamente adottato è "dropout" (Srivastava et al. 2014). Durante l'addestramento, dropout imposta casualmente una frazione p di output del nodo o attivazioni nascoste a zero. Questo incoraggia una maggiore distribuzione delle informazioni su più nodi anziché affidarsi a un piccolo numero di nodi. Al momento della previsione, viene utilizzata l'intera rete neurale, con attivazioni intermedie scalate di $1 - p$ per mantenere le ampiezze di output. Le ottimizzazioni GPU semplificano l'implementazione efficiente di dropout tramite framework come PyTorch e TensorFlow.
Siamo più precisi. Durante l'addestramento con dropout, l'output di ogni nodo a_i viene passato attraverso una maschera di dropout r_i prima di essere utilizzato dal layer successivo:

$$\tilde{a}_i = r_i \odot a_i$$

Dove:

- a_i - output del nodo i
- $\tilde{a}_i$ - output del nodo i dopo il dropout
- r_i - variabile casuale di Bernoulli indipendente con probabilità $1 - p$ di essere 1
- $\odot$ - moltiplicazione elemento per elemento

Per capire come funziona il dropout, è importante sapere che la maschera di dropout r_i è basata sulle variabili casuali di Bernoulli. Una variabile casuale di Bernoulli assume un valore di 1 con probabilità $1 - p$ (mantenendo l'attivazione) e un valore di 0 con probabilità p (dropping [perdendo] l'attivazione). Ciò significa che l'attivazione di ciascun nodo viene mantenuta o eliminata indipendentemente durante l'addestramento. Questa maschera di dropout r_i imposta casualmente una frazione p di attivazioni a 0 durante l'addestramento, costringendo la rete a creare rappresentazioni ridondanti.
Al momento del test, la maschera di dropout viene rimossa e le attivazioni vengono ridimensionate di $1 - p$ per mantenere le ampiezze di output previste:

$$a_i^{test} = (1 - p)a_i$$

Dove:

- a_i^{test} - output del nodo al momento del test
- p - la probabilità di effettuare il dropping [eliminare] di un nodo.

L'iperparametro chiave è p, la probabilità di eliminare ogni nodo, spesso impostata tra 0.2 e 0.5. Le reti più grandi tendono a trarre vantaggio da un dropout maggiore, mentre le reti più piccole rischiano di non adattarsi se vengono eliminati troppi nodi. Tentativi ed errori combinati con il monitoraggio delle prestazioni di validazione aiutano a regolare il livello di dropout.
Video 7.6 discute l'intuizione alla base della tecnica di regolarizzazione del dropout e il suo funzionamento.

Video 7.6: Dropout

https://www.youtube.com/watch?v=ARq74QuavAo&list=PLkDaE6sCZn6Hn0vK8co82zjQtt3T2Nkqc&i

7.7.3 Arresto Anticipato

L'intuizione alla base di "early stopping" arresto anticipato implica il monitoraggio delle prestazioni del modello su un set di validazione "held-out" [esterno] in epoche di

addestramento. Inizialmente, gli aumenti nell'idoneità del set di addestramento accompagnano i guadagni nell'accuratezza della validazione man mano che il modello rileva pattern generalizzabili. Dopo un certo punto, tuttavia, il modello inizia a sovradimensionarsi, agganciandosi a peculiarità e rumore nei dati di addestramento che non si applicano più in generale. Le prestazioni di validazione raggiungono il picco e poi si degradano se l'addestramento continua. Le regole di "arresto anticipato" interrompono l'addestramento a questo picco per evitare il sovradimensionamento. Questa tecnica dimostra come le pipeline ML debbano monitorare il feedback del sistema, non solo massimizzare incondizionatamente le prestazioni su un set di addestramento statico. Lo stato del sistema evolve e gli endpoint ottimali cambiano.
Pertanto, i metodi formali di arresto anticipato richiedono il monitoraggio di una metrica come l'accuratezza o la perdita di validazione dopo ogni epoca. Le curve comuni mostrano rapidi guadagni iniziali che si riducono gradualmente, alla fine raggiungendo un plateau e diminuiscono leggermente man mano che si verifica il sovradimensionamento. Il punto di arresto ottimale è spesso compreso tra 5 e 15 epoche oltre il picco, a seconda dei "patient threshold" [limiti della pazienza!]. Il monitoraggio di più metriche può migliorare il segnale poiché esiste una varianza tra le misure.
Le semplici regole di arresto anticipato si interrompono immediatamente alla prima degradazione post-picco. Metodi più robusti introducono un parametro di "pazienza", ovvero il numero di epoche di degradazione consentite prima dell'arresto. Ciò evita di interrompere prematuramente l'addestramento a causa di fluttuazioni transitorie. Le finestre di "pazienza" tipiche vanno da 50 a 200 batch di validazione. Finestre più ampie comportano il rischio di overfitting. Le strategie di ottimizzazione formali possono determinare la "pazienza" ottimale.

Esercizio 7.3: Regolarizzazione

Combattere l'Overfitting: Scoprire i Segreti della Regolarizzazione! L'overfitting è come se il modello memorizzasse le risposte a un test, per poi fallire l'esame reale. Le tecniche di regolarizzazione sono le guide di studio che aiutano il modello a generalizzare e ad affrontare nuovi problemi. In questo notebook Colab, impareremo come ottimizzare i parametri di regolarizzazione per risultati ottimali utilizzando la regolarizzazione L1 e L2, il dropout e l'arresto anticipato.

Open in Colab

Video 7.7 tratta alcuni altri metodi di regolarizzazione che possono ridurre l'overfitting del modello.

Video 7.7: Altri Metodi di Regolarizzazione

https://www.youtube.com/watch?v=BOCLq2gpcGU&list=PLkDaE6sCZn6Hn0vK8co82zjQtt3T2Nkqc&index=8

7.8 Funzioni di Attivazione

Le funzioni di attivazione svolgono un ruolo cruciale nelle reti neurali. Introducono comportamenti non lineari che consentono alle reti neurali di modellare pattern complessi. Le funzioni di attivazione elemento per elemento vengono applicate alle somme ponderate che arrivano a ciascun neurone nella rete. Senza funzioni di attivazione, le reti neurali sarebbero ridotte a modelli di regressione lineare.
Idealmente, le funzioni di attivazione possiedono alcune qualità desiderabili:

- **Non lineari:** Consentono di modellare relazioni complesse tramite trasformazioni non lineari della somma degli input.

- **Differenziabili:** Devono avere derivate prime ben definite per abilitare la retropropagazione e l'ottimizzazione basata sul gradiente durante l'addestramento.
- **Limitazione dell'Intervallo:** Limitano il segnale di output, impedendo un'esplosione. Ad esempio, la sigmoide schiaccia gli input a (0,1).

Inoltre, proprietà come efficienza computazionale, monotonicità e fluidità rendono alcune attivazioni più adatte di altre in base all'architettura di rete e alla complessità del problema. Esamineremo brevemente alcune delle funzioni di attivazione più ampiamente adottate e i loro punti di forza e limiti. Forniremo anche linee guida per la selezione di funzioni appropriate abbinate ai vincoli del sistema ML e alle esigenze dei casi d'uso.

7.8.1 Sigmoide

L'attivazione sigmoide applica una curva a forma di S schiacciante che lega strettamente l'output tra 0 e 1. Ha la forma matematica:

$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

La trasformazione esponenziale consente alla funzione di passare gradualmente da quasi 0 a quasi 1 quando l'input passa da molto negativo a molto positivo. L'aumento monotono copre l'intero intervallo (0,1).

La funzione sigmoide presenta diversi vantaggi. Fornisce sempre un gradiente uniforme per la retropropagazione e il suo output è limitato tra 0 e 1, il che aiuta a prevenire valori "esplosivi" durante l'addestramento. Inoltre, ha una semplice formula matematica che è facile da calcolare.

Tuttavia, la funzione sigmoide presenta anche alcuni svantaggi. Tende a saturarsi a valori di input estremi, il che può causare la "scomparsa" dei gradienti, rallentando o addirittura interrompendo il processo di apprendimento. Inoltre, la funzione non è centrata sullo zero, il che significa che i suoi output non sono distribuiti simmetricamente attorno allo zero, il che può portare ad aggiornamenti inefficienti durante l'addestramento.

7.8.2 Tanh

Anche Tanh o "tangente iperbolica" assume una forma a S ma è centrata sullo zero, il che significa che il valore medio dell'output è 0.

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

La trasformazione numeratore/denominatore sposta l'intervallo da (0,1) in Sigmoide a (-1, 1) in tanh.

La maggior parte dei pro/contro sono condivisi con la Sigmoide, ma Tanh evita alcuni problemi di saturazione dell'output essendo centrata. Tuttavia, soffre ancora di gradienti che svaniscono con molti layer.

7.8.3 ReLU

La Rectified Linear Unit (ReLU) introduce un semplice comportamento di soglia con la sua forma matematica:

$$ReLU(x) = max(0, x)$$

Lascia tutti gli input positivi invariati mentre taglia tutti i valori negativi a 0. Questa attivazione sparsa e il calcolo economico rendono ReLU ampiamente favorito rispetto a sigmoide/tanh. Figura 7.6 dimostra le 3 funzioni di attivazione di cui abbiamo discusso sopra in confronto a una funzione lineare:

Figura 7.6: Funzioni di Attivazione Comuni. Fonte: AI Wiki.

7.8.4 Softmax

La funzione di attivazione softmax è generalmente utilizzata come ultimo layer per le attività di classificazione per normalizzare il vettore del valore di attivazione in modo che i suoi elementi sommino a 1. Questo è utile per le attività di classificazione in cui vogliamo imparare a prevedere probabilità specifiche per classe di un input particolare, nel qual caso la probabilità cumulativa tra le classi è uguale a 1. La funzione di attivazione softmax è definita come

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1,2, \dots, K$$

7.8.5 Pro e Contro

Tabella 7.4 sono i pro e i contro riassuntivi di queste varie funzioni di attivazione standard:

Tabella 7.4: Confronto dei pro e dei contro di diversi algoritmi di ottimizzazione.

Attivazione	Pro	Contro
Sigmoide	• Gradiente uniforme per il backdrop [sfondo] • Output limitato tra 0 e 1	• La saturazione elimina i gradienti • Non centrato sullo zero
Tanh	• Gradiente più uniforme della sigmoide • Output centrato sullo zero [-1, 1]	• Soffre ancora di problemi di gradiente evanescente
ReLU	• Efficiente dal punto di vista computazionale • Introduce la “sparsity” [scarsità] • Evita gradienti evanescenti	• Unità “ReLU morenti” • Non limitato
Softmax	• Utilizzato per l’ultimo livello per normalizzare gli output in modo che siano una distribuzione	

- In genere utilizzato per attività di classificazione

Esercizio 7.4: Funzioni di Attivazione

Sblocchiamo la potenza delle funzioni di attivazione! Questi piccoli "muletti" matematici sono ciò che rende le reti neurali così incredibilmente flessibili. In questo notebook Colab, ci si cimenterà con funzioni come Sigmoid, tanh e la superstar ReLU. Guardiamo come trasformano gli input e scopriamo quale funziona meglio in diverse situazioni. È la chiave per costruire reti neurali in grado di affrontare problemi complessi!

Open in Colab

7.9 Inizializzazione dei Pesi

La corretta inizializzazione dei pesi in una rete neurale prima dell'addestramento è un passaggio fondamentale che ha un impatto diretto sulle prestazioni del modello. L'inizializzazione casuale dei pesi a valori molto grandi o molto piccoli può portare a problemi come gradienti che svaniscono/esplodono, convergenza lenta dell'addestramento o intrappolati in minimi locali scadenti. La corretta inizializzazione del peso accelera la convergenza del modello durante l'addestramento e comporta implicazioni per le prestazioni del sistema al momento dell'inferenza negli ambienti di produzione. Alcuni aspetti chiave sono:

- **Tempo di Accuratezza più Rapido:** Un'inizializzazione attentamente calibrata porta a una convergenza più rapida, che si traduce nel raggiungimento da parte dei modelli di traguardi di accuratezza target in anticipo nel ciclo di training. Ad esempio, l'inizializzazione Xavier potrebbe ridurre il tempo di accuratezza del 20% rispetto a un'inizializzazione casuale errata. Poiché l'addestramento è in genere la fase più dispendiosa in termini di tempo e calcolo, ciò migliora direttamente la velocità e la produttività del sistema ML.
- **Efficienza del Ciclo di Iterazione del Modello:** Se i modelli vengono addestrati più rapidamente, il tempo di risposta complessivo per le iterazioni di sperimentazione, valutazione e progettazione del modello diminuisce in modo significativo. I sistemi hanno maggiore flessibilità per esplorare architetture, pipeline di dati, ecc., entro determinati intervalli di tempo.
- **Impatto sulle Epoche di Addestramento Necessarie:** Il processo di addestramento viene eseguito per più epoche, con ogni passaggio completo attraverso i dati che rappresenta un'epoca. Una buona inizializzazione può ridurre le epoche necessarie per far convergere le curve di perdita e accuratezza sul set di addestramento del 10-30%. Ciò significa risparmi tangibili sui costi di risorse e infrastruttura.
- **Effetto sugli Iperparametri di Addestramento:** I parametri di inizializzazione del peso interagiscono fortemente con determinati iperparametri di regolarizzazione che governano le dinamiche di addestramento, come i programmi di velocità di apprendimento e le probabilità di abbandono. Trovare la giusta combinazione di impostazioni non è banale. Un'inizializzazione appropriata semplifica questa ricerca.

L'inizializzazione dei pesi ha vantaggi a cascata per l'efficienza ingegneristica dell'apprendimento automatico e un overhead di risorse di sistema ridotto al minimo. È una tattica facilmente trascurata che ogni professionista dovrebbe padroneggiare. La scelta di quale tecnica di inizializzazione del peso utilizzare dipende da fattori come l'architettura del modello (numero di layer, pattern di connettività, ecc.), le funzioni di attivazione e il problema specifico da risolvere. Nel corso degli anni, i ricercatori hanno sviluppato e verificato

empiricamente diverse strategie di inizializzazione mirate alle comuni architetture di reti neurali, di cui parleremo qui.

7.9.1 Inizializzazione Uniforme e Normale

Quando si inizializzano pesi in modo casuale, vengono comunemente utilizzate due distribuzioni di probabilità standard: uniforme e Gaussiana (normale). La distribuzione uniforme imposta una probabilità uguale che i parametri di peso iniziali rientrino in qualsiasi punto entro i limiti minimi e massimi impostati. Ad esempio, i limiti potrebbero essere -1 e 1, portando a una distribuzione uniforme dei pesi tra questi limiti. La distribuzione gaussiana, d'altra parte, concentra la probabilità attorno a un valore medio, seguendo la forma di una curva a campana. La maggior parte dei valori di peso si raggrupperà nella regione della media specificata, con meno campioni verso le estremità. Il parametro di deviazione standard controlla la distribuzione attorno alla media.

La scelta tra inizializzazione uniforme o normale dipende dall'architettura di rete e dalle funzioni di attivazione. Per reti poco profonde, si consiglia una distribuzione normale con una deviazione standard relativamente piccola (ad esempio, 0.01). La curva a campana impedisce valori di peso elevati che potrebbero innescare l'instabilità di addestramento in reti piccole. Per reti più profonde, una distribuzione normale con deviazione standard più elevata (diciamo 0.5 o superiore) o una distribuzione uniforme può essere preferita per tenere conto dei problemi di gradiente evanescente su molti layer. La maggiore diffusione determina una maggiore differenziazione tra i comportamenti dei neuroni. La messa a punto dei parametri di distribuzione di inizializzazione è fondamentale per una convergenza stabile e rapida del modello. Il monitoraggio dei trend di "loss" [perdita] di addestramento può diagnosticare i problemi per modificare i parametri in modo iterativo.

7.9.2 Inizializzazione Xavier

Proposta da Glorot e Bengio (2010), questa tecnica di inizializzazione è progettata appositamente per le funzioni di attivazione sigmoide e tanh. Queste attivazioni saturate possono causare gradienti evanescenti o esplosivi durante la retro-propagazione su molti layer.

Il metodo Xavier imposta in modo intelligente la varianza della distribuzione dei pesi in base al numero di input e output per ciascun layer. L'intuizione è che questo bilancia il flusso di informazioni e gradienti in tutta la rete. Ad esempio, si consideri un layer con 300 unità di input e 100 unità di output. Inserendo questo nella formula varianza = 2/(#inputs + #outputs) si ottiene una varianza di 2/(300+100) = 0.01.

Il campionamento dei pesi iniziali da una distribuzione uniforme o normale centrata su 0 con questa varianza fornisce una convergenza di addestramento molto più fluida per reti sigmoide/tanh profonde. I gradienti sono ben condizionati, impedendo la scomparsa o la crescita esponenziale.

7.9.3 Inizializzazione He

Come proposto da K. He et al. (2015), questa tecnica di inizializzazione è adattata alle funzioni di attivazione ReLU (Rectified Linear Unit). Le ReLU introducono il problema del neurone morente in cui le unità rimangono bloccate e producono solo 0 se inizialmente ricevono forti input negativi. Ciò rallenta e ostacola l'addestramento.

"He" supera questo problema campionando i pesi da una distribuzione con un set di varianza basato solo sul numero di input per layer, ignorando gli output. Ciò mantiene i segnali in arrivo sufficientemente piccoli da attivare le ReLU nel loro regime lineare dall'inizio, evitando unità morte. Per un layer con 1024 input, la formula varianza = 2/1024 = 0.002 mantiene la maggior parte dei pesi concentrati strettamente attorno a 0.

Questa inizializzazione specializzata consente alle reti ReLU di convergere in modo efficiente fin dall'inizio. La scelta tra Xavier e He deve corrispondere alla funzione di attivazione della rete prevista.

Esercizio 7.5: Inizializzazione dei Pesi

Facciamo partire la rete neurale col piede giusto con l'inizializzazione dei pesi! Il modo in cui si impostano quei pesi iniziali può fare la differenza nell'addestramento del modello. Si immagini di accordare gli strumenti di un'orchestra prima del concerto. In questo notebook Colab, si imparerà che la giusta strategia di inizializzazione può far risparmiare tempo, migliorare le prestazioni del modello e rendere il percorso di deep-learning molto più fluido.

Open in Colab

Video 7.8 sottolinea l'importanza di selezionare deliberatamente i valori di peso iniziale rispetto a scelte casuali.

Video 7.8: Inizializzazione dei Pesi

https://www.youtube.com/watch?v=s2coXdufOzE&list=PLkDaE6sCZn6Hn0vK8co82zjQtt3T2Nkqc&in

7.10 "Colli di Bottiglia" del Sistema

Come introdotto in precedenza, le reti neurali comprendono operazioni lineari (moltiplicazioni di matrici) intervallate da funzioni di attivazione non lineari elemento per elemento. La parte computazionalmente più costosa delle reti neurali sono le trasformazioni lineari, in particolare le moltiplicazioni di matrici tra ogni layer. Questi layer lineari mappano le attivazioni dal layer precedente a uno spazio dimensionale superiore che funge da input per la funzione di attivazione del layer successivo.

7.10.1 Complessità a Runtime della Moltiplicazione di Matrici

7.10.1.1 Moltiplicazioni di Layer vs. Attivazioni

La maggior parte del calcolo nelle reti neurali deriva dalle moltiplicazioni di matrici tra layer. Si consideri un layer di rete neurale con una dimensione di input di M = 500 e una dimensione di output di N = 1000; la moltiplicazione di matrici richiede $O(N \cdot M) = O(1000 \cdot 500) = 500{,}000$ operazioni di moltiplicazione-accumulazione (MAC) tra quei layer.
Confronta questo col layer precedente, che aveva M = 300 input, che richiedevano $O(500 \cdot 300) = 150{,}000$ operazioni. Man mano che le dimensioni dei layer aumentano, i requisiti computazionali aumentano quadraticamente con la dimensione del layer. I calcoli totali su L layer possono essere espressi come $\sum_{l=1}^{L-1} O\left(N^{(l)} \cdot M^{(l-1)}\right)$, dove il calcolo richiesto per ogni layer dipende dal prodotto delle dimensioni di input e output delle matrici che vengono moltiplicate.
Ora, confrontando la moltiplicazione della matrice con la funzione di attivazione, che richiede solo $O(N) = 1000$ non linearità elemento per elemento per $N = 1000$ output, possiamo vedere le trasformazioni lineari che dominano le attivazioni computazionalmente.
Queste grandi moltiplicazioni di matrici influiscono sulle scelte hardware, sulla latenza dell'inferenza e sui vincoli di potenza per le applicazioni di reti neurali nel mondo reale. Ad esempio, un tipico layer DNN potrebbe richiedere 500,000 moltiplicazioni-accumulazioni rispetto a solo 1000 attivazioni non lineari, dimostrando un aumento di 500x nelle operazioni matematiche.
Quando si addestrano reti neurali, in genere utilizziamo la discesa del gradiente in mini-batch, operando su piccoli batch di dati contemporaneamente. Considerando una

dimensione batch di B esempi di addestramento, l'input per la moltiplicazione di matrice diventa una matrice $M \times B$, mentre l'output è una matrice $N \times B$.

7.10.1.2 Mini-batch

Nell'addestramento delle reti neurali, dobbiamo stimare ripetutamente il gradiente della funzione di perdita rispetto ai parametri di rete (ad esempio, pesi e bias). Questo gradiente indica in quale direzione i parametri devono essere aggiornati per ridurre al minimo la perdita. Come introdotto in precedenza, eseguiamo aggiornamenti su un batch di dati a ogni aggiornamento, noto anche come discesa del gradiente stocastico o "discesa del gradiente mini-batch".

L'approccio più semplice è stimare il gradiente in base a un singolo esempio di addestramento, calcolare l'aggiornamento dei parametri, riassettare tutto e ripetere per l'esempio successivo. Tuttavia, ciò comporta aggiornamenti dei parametri molto piccoli e frequenti che possono essere computazionalmente inefficienti e potrebbero dover essere più accurati in termini di convergenza a causa della stocasticità dell'utilizzo di un solo dato per un aggiornamento del modello.

Invece, la discesa del gradiente in mini-batch bilancia la stabilità della convergenza e l'efficienza computazionale. Invece di calcolare il gradiente su singoli esempi, stimiamo il gradiente in base a piccoli "mini-batch" di dati, solitamente tra 8 e 256 esempi in pratica. Ciò fornisce una stima del gradiente rumorosa ma coerente che porta a una convergenza più stabile. Inoltre, l'aggiornamento dei parametri deve essere eseguito solo una volta per mini-batch anziché una volta per ogni esempio, riducendo il sovraccarico computazionale.

Regolando la dimensione del mini-batch, possiamo controllare il compromesso tra la fluidità della stima (i batch più grandi sono generalmente migliori) e la frequenza degli aggiornamenti (i batch più piccoli consentono aggiornamenti più frequenti). Le dimensioni del mini-batch sono solitamente potenze di 2, quindi possono sfruttare in modo efficiente il parallelismo tra i core GPU.

Quindi, il calcolo totale esegue una moltiplicazione di matrici $N \times M$ per $M \times B$, producendo $O(N \cdot M \cdot B)$ operazioni in virgola mobile. Come esempio numerico, $N = 1000$ unità nascoste, $M = 500$ unità di input e una dimensione del batch $B = 64$ equivale a 1000 x 500 x 64 = 32 milioni di moltiplicazioni-accumulazioni per iterazione di training!

Al contrario, le funzioni di attivazione vengono applicate elemento per elemento alla matrice di output $N \times B$, richiedendo solo $O(N \cdot B)$ calcoli. Per $N = 1000$ e $B = 64$, si tratta di sole 64,000 non linearità, ovvero 500 volte meno lavoro della moltiplicazione di matrici.

Man mano che aumentiamo le dimensioni del batch per sfruttare appieno hardware parallelo come le GPU, la discrepanza tra la moltiplicazione di matrici e il costo della funzione di attivazione aumenta ulteriormente. Ciò rivela come l'ottimizzazione delle operazioni di algebra lineare offra enormi guadagni di efficienza.

Pertanto, la moltiplicazione di matrici è fondamentale nell'analisi di dove e come le reti neurali impiegano i calcoli. Ad esempio, le moltiplicazioni di matrici spesso rappresentano oltre il 90% della latenza di inferenza e del tempo di addestramento nelle comuni reti neurali convoluzionali e ricorrenti.

7.10.1.3 Ottimizzazione della Moltiplicazione di Matrici

Diverse tecniche migliorano l'efficienza delle operazioni generali matrice-matrice densa/sparsa e matrice-vettore per migliorare l'efficienza complessiva. Alcuni metodi chiave sono:

- Sfruttamento di librerie matematiche ottimizzate come cuBLAS per l'accelerazione GPU

- Abilitazione di formati di precisione inferiore come FP16 o INT8 dove l'accuratezza lo consente
- Utilizzo di Tensor Processing Unit con moltiplicazione di matrici in hardware
- Calcoli consapevoli della sparsità e formati di archiviazione dati per sfruttare i parametri zero
- Approssimazione delle moltiplicazioni di matrici con algoritmi come le Fast Fourier Transform
- Progettazione dell'architettura del modello per ridurre le larghezze e le attivazioni degli layer
- Quantizzazione, pruning [potatura], distillazione e altre tecniche di compressione
- Parallelizzazione del calcolo sull'hardware disponibile
- Risultati di caching/pre-calcolo ove possibile per ridurre le operazioni ridondanti

Le potenziali tecniche di ottimizzazione sono vaste, data la porzione sproporzionata di tempo che i modelli trascorrono nella matematica di matrici e vettori. Anche i miglioramenti incrementali velocizzano i tempi di esecuzione e riducono il consumo di energia. Trovare nuovi modi per migliorare queste primitive di algebra lineare rimane un'area di ricerca attiva allineata con le future esigenze di machine learning. Ne parleremo in dettaglio nei capitoli Ottimizzazioni e Accelerazione IA.

7.10.2 Calcolo vs. Collo di Bottiglia della Memoria

A questo punto, la moltiplicazione matrice-matrice è l'operazione matematica fondamentale alla base delle reti neurali. Sia l'addestramento che l'inferenza per le reti neurali utilizzano ampiamente queste operazioni di moltiplicazione di matrici. L'analisi mostra che oltre il 90% dei requisiti computazionali nelle reti neurali attuali derivano da moltiplicazioni di matrici. Di conseguenza, le prestazioni della moltiplicazione di matrici hanno un'enorme influenza sul tempo complessivo di addestramento o inferenza del modello.

7.10.2.1 Addestramento vs. Inferenza

Mentre l'addestramento e l'inferenza si basano ampiamente sulle prestazioni della moltiplicazione di matrici, i loro profili computazionali precisi differiscono. In particolare, l'inferenza della rete neurale tende a essere più legata al calcolo rispetto all'addestramento per una dimensione di batch equivalente. La differenza fondamentale risiede nel passaggio di backpropagation, che è richiesto solo durante l'addestramento. La backpropagation implica una sequenza di operazioni di moltiplicazione di matrici per calcolare i gradienti rispetto alle attivazioni su ogni layer della rete. Tuttavia, è fondamentale che qui non sia necessaria alcuna larghezza di banda di memoria aggiuntiva: gli input, gli output e i gradienti vengono letti/scritti dalla cache o dai registri.

Di conseguenza, l'addestramento mostra intensità aritmetiche inferiori, con calcoli del gradiente limitati dall'accesso alla memoria anziché dai **FLOP** (Floating Point Operations Per Second), una misura delle prestazioni computazionali che indica quanti calcoli in virgola mobile un sistema può eseguire al secondo. Al contrario, la propagazione in avanti domina l'inferenza della rete neurale, che corrisponde a una serie di moltiplicazioni matrice-matrice. Senza una retrospettiva del gradiente che richiede molta memoria, le dimensioni dei batch più grandi spingono facilmente l'inferenza a essere estremamente limitata dal calcolo. Le elevate intensità aritmetiche misurate mostrano questo. I tempi di risposta possono essere critici per alcune applicazioni di inferenza, costringendo il fornitore dell'applicazione a utilizzare una dimensione di batch inferiore per soddisfare questi requisiti di tempo di

risposta, riducendo così l'efficienza dell'hardware; quindi, le inferenze potrebbero vedere un utilizzo inferiore dell'hardware.
Le implicazioni sono che il provisioning hardware e i compromessi tra larghezza di banda e FLOP differiscono a seconda che un sistema miri al training o all'inferenza. I server ad alta produttività e bassa latenza per l'inferenza dovrebbero enfatizzare la potenza di calcolo anziché la memoria, mentre i cluster di training richiedono un'architettura più bilanciata.
Tuttavia, la moltiplicazione di matrici mostra un'interessante tensione: la larghezza di banda della memoria dell'hardware sottostante o le capacità di throughput aritmetico possono vincolarla. La capacità del sistema di recuperare e fornire dati matriciali rispetto alla sua capacità di eseguire operazioni di calcolo determina questa direzione.
Questo fenomeno ha impatti profondi; l'hardware deve essere progettato giudiziosamente e devono essere prese in considerazione le ottimizzazioni del software. Ottimizzare e bilanciare il calcolo rispetto alla memoria per alleviare questo collo di bottiglia della moltiplicazione di matrici è fondamentale per un training un deployment efficienti del modello.
Infine, la dimensione del batch può avere un impatto sui tassi di convergenza durante l'addestramento della rete neurale, un'altra considerazione importante. Ad esempio, ci sono generalmente rendimenti decrescenti nei benefici della convergenza con dimensioni di batch estremamente grandi (ad esempio, > 16384). Al contrario, dimensioni di batch estremamente grandi possono essere sempre più vantaggiose da una prospettiva di intensità hardware/aritmetica; l'utilizzo di batch così grandi potrebbe non tradursi in una convergenza più rapida rispetto al tempo a causa dei loro benefici decrescenti per la convergenza. Questi compromessi fanno parte delle decisioni di progettazione fondamentali per i sistemi per il tipo di ricerca basata sull'apprendimento automatico.

7.10.2.2 Dimensione del Batch

La dimensione del batch utilizzata durante l'addestramento e l'inferenza della rete neurale ha un impatto significativo sul fatto che la moltiplicazione di matrici rappresenti un collo di bottiglia computazionale o di memoria. In concreto, la dimensione del batch si riferisce al numero di campioni propagati assieme attraverso la rete in un passaggio avanti/indietro. La moltiplicazione di matrici equivale a dimensioni di matrice maggiori.
In particolare, diamo un'occhiata all'intensità aritmetica della moltiplicazione di matrici durante l'addestramento della rete neurale. Questa misura il rapporto tra operazioni computazionali e trasferimenti di memoria. La moltiplicazione di due matrici di dimensione $N \times M$ e $M \times B$ richiede $N \times M \times B$ operazioni di moltiplicazione-accumulo, ma solo trasferimenti di $N \times M + M \times B$ elementi di matrice.
Man mano che aumentiamo la dimensione del batch B, il numero di operazioni aritmetiche cresce più velocemente dei trasferimenti di memoria. Ad esempio, con una dimensione del batch di 1, abbiamo bisogno di $N \times M$ operazioni e $N + M$ trasferimenti, dando un rapporto di intensità aritmetica di circa $\frac{N \times M}{N+M}$. Ma con una dimensione del batch di grandi dimensioni di 128, il rapporto di intensità diventa $\frac{128 \times N \times M}{N \times M + M \times 128} \approx 128$.
L'utilizzo di una dimensione del batch più grande sposta il calcolo complessivo da vincolato alla memoria a più vincolato al calcolo. L'addestramento IA utilizza grandi dimensioni del batch ed è generalmente limitato dalle massime prestazioni di calcolo aritmetiche, ovvero l'Applicazione 3 in Figura 7.7. Pertanto, la moltiplicazione di matrici in batch è molto più intensiva dal punto di vista computazionale rispetto al limite di accesso alla memoria. Ciò ha implicazioni per la progettazione hardware e le ottimizzazioni software, che tratteremo in seguito. L'intuizione chiave è che possiamo modificare in modo significativo il profilo computazionale e i colli di bottiglia posti dall'addestramento e dall'inferenza della rete neurale regolando la dimensione del batch.

Figura 7.7: Modello a profilo a di tetto per il training di IA.

7.10.2.3 Caratteristiche Hardware

L'hardware moderno come CPU e GPU è altamente ottimizzato per la produttività computazionale piuttosto che per la larghezza di banda della memoria. Ad esempio, le GPU H100 Tensor Core di fascia alta possono fornire oltre 60 TFLOPS di prestazioni a doppia precisione, ma forniscono solo fino a 3 TB/s di larghezza di banda della memoria. Ciò significa che c'è uno squilibrio di quasi 20 volte tra unità aritmetiche e accesso alla memoria; di conseguenza, per hardware come gli acceleratori GPU, i carichi di lavoro di addestramento della rete neurale devono essere resi il più intensivi possibile dal punto di vista computazionale per utilizzare appieno le risorse disponibili.

Ciò motiva ulteriormente la necessità di utilizzare batch di grandi dimensioni durante l'addestramento. Quando si utilizza un batch di piccole dimensioni, la moltiplicazione della matrice è limitata dalla larghezza di banda della memoria, sottoutilizzando le abbondanti risorse di elaborazione. Tuttavia, possiamo spostare il collo di bottiglia verso l'elaborazione e ottenere un'intensità aritmetica molto più elevata con batch sufficientemente grandi. Ad esempio, potrebbero essere necessari batch di 256 o 512 campioni per saturare una GPU di fascia alta. Lo svantaggio è che batch più grandi forniscono aggiornamenti dei parametri meno frequenti, il che può influire sulla convergenza. Tuttavia, il parametro funge da importante manopola di sintonizzazione per bilanciare le limitazioni di memoria e quelle di elaborazione.

Pertanto, date le architetture di elaborazione-memoria sbilanciate dell'hardware moderno, l'impiego di batch di grandi dimensioni è essenziale per alleviare i colli di bottiglia e massimizzare la produttività. Come accennato, anche il software e gli algoritmi successivi devono adattarsi a tali dimensioni di batch, poiché dimensioni di batch più grandi possono avere rendimenti decrescenti verso la convergenza della rete. L'utilizzo di dimensioni di batch molto piccole può portare a un utilizzo non ottimale dell'hardware, limitando in ultima analisi l'efficienza del training. L'aumento di dimensioni dei batch di grandi dimensioni è un argomento di ricerca esplorato in vari lavori che mirano a eseguire una training su larga scala (Y. You et al. 2017).

7.10.2.4 Architetture dei Modelli

L'architettura della rete neurale influisce anche sul fatto che la moltiplicazione di matrici rappresenti un collo di bottiglia computazionale o di memoria maggiore durante l'esecuzione. I trasformatori e gli MLP sono molto più vincolati al calcolo rispetto alle reti neurali convoluzionali CNN. Ciò deriva dai tipi di operazioni di moltiplicazione di matrici coinvolte in ciascun modello. I trasformatori si basano sull'auto-attenzione, moltiplicando grandi matrici di attivazione per enormi matrici di parametri per correlare gli elementi. Gli MLP impilano layer completamente connessi, richiedendo anche grandi moltiplicazioni matriciali.
Al contrario, i layer convoluzionali nelle CNN hanno una finestra scorrevole che riutilizza attivazioni e parametri nell'input, il che significa che sono necessarie meno operazioni matriciali univoche. Tuttavia, le convoluzioni richiedono l'accesso ripetuto a piccole parti di input e lo spostamento di somme parziali per popolare ciascuna finestra. Sebbene le operazioni aritmetiche nelle convoluzioni siano intense, questo spostamento di dati e la manipolazione del buffer impongono enormi overhead di accesso alla memoria. Le CNN comprendono diverse fasi a strati, quindi gli output intermedi devono materializzarsi frequentemente nella memoria.
Di conseguenza, l'addestramento CNN tende a essere più vincolato alla larghezza di banda della memoria rispetto al limite aritmetico in confronto a Transformers e MLP. Pertanto, il profilo di moltiplicazione della matrice e, a sua volta, il collo di bottiglia posto, varia in modo significativo in base alla scelta del modello. Hardware e sistemi devono essere progettati con un appropriato equilibrio di larghezza di banda di elaborazione-memoria a seconda dell'implementazione del modello target. I modelli che si basano maggiormente sull'attenzione e sui layer MLP richiedono una maggiore produttività aritmetica rispetto alle CNN, il che richiede un'elevata larghezza di banda della memoria.

7.11 Parallelizzazione del Training

L'addestramento delle reti neurali comporta richieste di calcolo e memoria intensive. L'algoritmo di backpropagation per il calcolo dei gradienti e l'aggiornamento dei pesi consiste in ripetute moltiplicazioni di matrici e operazioni aritmetiche sull'intero set di dati. Ad esempio, un passaggio di backpropagation scala in complessità temporale con $O(num_parameters \times batch_size \times sequence_length)$.
I requisiti di calcolo aumentano rapidamente con l'aumento delle dimensioni del modello in parametri e layer. Inoltre, l'algoritmo richiede l'archiviazione di output di attivazione e parametri del modello per la fase di backward, che cresce con le dimensioni del modello.
I modelli più grandi non possono adattarsi e addestrarsi su un singolo dispositivo acceleratore come una GPU e l'ingombro di memoria diventa proibitivo. Pertanto, dobbiamo parallelizzare l'addestramento del modello su più dispositivi per fornire elaborazione e memoria sufficienti per addestrare reti neurali all'avanguardia.
Come mostrato in Figura 7.8, i due approcci principali sono il parallelismo dei dati, che replica il modello su più dispositivi suddividendo i dati di input in batch, e il parallelismo del modello, che suddivide l'architettura del modello stesso su diversi dispositivi. Tramite il training in parallelo, possiamo sfruttare maggiori risorse aggregate di elaborazione e memoria per superare le limitazioni del sistema e accelerare i carichi di lavoro di deep learning.

Figura 7.8: Parallelismo dei dati e parallelismo del modello.

7.11.1 Parallelismo dei Dati

La parallelizzazione dei dati è un approccio comune per parallelizzare il training di apprendimento automatico su più unità di elaborazione, come GPU o risorse di elaborazione distribuite. Il set di dati di addestramento è suddiviso in batch nel parallelismo dei dati e un'unità di elaborazione separata elabora ogni batch. I parametri del modello vengono poi aggiornati in base ai gradienti calcolati dall'elaborazione di ogni batch. Ecco una descrizione dettagliata della parallelizzazione dei dati per il training ML:

1. **Divisione del Dataset:** Il set di dati di addestramento è suddiviso in batch più piccoli, ciascuno contenente un sottoinsieme degli esempi di training.
2. **Replica del Modello:** Il modello di rete neurale è replicato su tutte le unità di elaborazione e ogni unità di elaborazione ha la sua copia.
3. **Calcolo Parallelo:** Ogni unità di elaborazione prende un batch diverso e calcola in modo indipendente i passaggi in forward e backward. Durante il passaggio forward [in avanti], il modello fa delle previsioni sui dati di input. La funzione di loss [perdita] determina i gradienti per i parametri del modello durante il passaggio backward [all'indietro].
4. **Aggregazione dei Gradienti:** Dopo l'elaborazione dei rispettivi batch, i gradienti di ogni unità di elaborazione vengono aggregati. I metodi di aggregazione comuni includono la sommatoria o la media dei gradienti.
5. **Aggiornamento dei Parametri:** I gradienti aggregati aggiornano i parametri del modello. L'aggiornamento può essere eseguito utilizzando algoritmi di ottimizzazione come SGD o varianti come Adam.
6. **Sincronizzazione:** Dopo l'aggiornamento, tutte le unità di elaborazione sincronizzano i parametri del modello, assicurandosi che ciascuna ne abbia la versione più recente.

I passaggi precedenti vengono ripetuti per diverse iterazioni o fino alla convergenza. Prendiamo un esempio specifico. Abbiamo 256 dimensioni di batch e 8 GPU; ogni GPU riceverà un micro-batch di 32 campioni. I loro passaggi forward e backward calcolano perdite e gradienti solo in base ai 32 campioni locali. I gradienti vengono aggregati tra i dispositivi con un server dei parametri o una libreria di comunicazioni collettiva per ottenere il gradiente effettivo per il batch globale. Gli aggiornamenti dei pesi avvengono indipendentemente su ogni GPU in base a questi gradienti. Dopo un numero configurato di iterazioni, i pesi

aggiornati si sincronizzano e si equalizzano tra i dispositivi prima di passare alle iterazioni successive.
Il parallelismo dei dati è efficace quando il modello è grande e il set di dati è sostanziale, poiché consente l'elaborazione parallela di diverse parti dei dati. È ampiamente utilizzato in framework e librerie di deep learning che supportano il training distribuito, come TensorFlow e PyTorch. Tuttavia, per garantire una parallelizzazione efficiente, è necessario prestare attenzione a gestire problemi come l sovraccarico della comunicazione, bilanciamento del carico e sincronizzazione.

7.11.2 Parallelismo del Modello

Il parallelismo del modello si riferisce alla distribuzione del modello di rete neurale su più dispositivi anziché alla replica del modello completo come il parallelismo dei dati. Ciò è particolarmente utile quando un modello è troppo grande per essere inserito nella memoria di una singola GPU o di un dispositivo acceleratore. Sebbene ciò potrebbe non essere specificamente applicabile per casi d'uso embedded o TinyML poiché la maggior parte dei modelli è relativamente piccola, è comunque utile saperlo.
Nell'addestramento parallelo del modello, diverse parti o layer del modello vengono assegnati a dispositivi separati. Le attivazioni di input e gli output intermedi vengono partizionati e passati tra questi dispositivi durante i passaggi forward e backward per coordinare i calcoli del gradiente tra le partizioni del modello.
Il "footprint" [impronta] di memoria e le operazioni di calcolo vengono distribuite suddividendo l'architettura del modello su più dispositivi anziché concentrarsi su uno. Ciò consente l'addestramento di modelli molto grandi con miliardi di parametri che altrimenti supererebbero la capacità di un singolo dispositivo. Esistono diversi modi in cui possiamo eseguire il partizionamento:

- **Parallelismo di Layer:** I layer consecutivi sono distribuiti su dispositivi diversi. Ad esempio, il dispositivo 1 contiene i layer 1-3; il dispositivo 2 contiene i layer 4-6. Le attivazioni di output dal layer 3 verrebbero trasferite al dispositivo 2 per avviare i layer successivi per i calcoli della fase di forward.
- **Parallelismo a Livello di Filtro:** Nei layer convoluzionali, i filtri di output possono essere suddivisi tra più dispositivi. Ogni dispositivo calcola gli output di attivazione per un sottoinsieme di filtri, che vengono concatenati prima di propagarsi ulteriormente.
- **Parallelismo Spaziale:** Le immagini di input vengono divise spazialmente, quindi ogni dispositivo elabora una determinata regione come il quarto in alto a sinistra delle immagini. Le regioni di output si combinano poi per formare l'output completo.

Inoltre, le combinazioni ibride possono suddividere il modello a livello di layer e i dati in batch. Il tipo appropriato di parallelismo del modello dipende dai vincoli specifici dell'architettura neurale e dalla configurazione hardware. Ottimizzare il partizionamento e la comunicazione per la topologia del modello è fondamentale per ridurre al minimo il sovraccarico.
Tuttavia, poiché le parti del modello vengono eseguite su dispositivi fisicamente separati, devono comunicare e sincronizzare i loro parametri durante ogni fase di addestramento. La fase di backward deve garantire che gli aggiornamenti del gradiente si propaghino accuratamente tra le partizioni del modello. Quindi, il coordinamento e l'interconnessione ad alta velocità tra i dispositivi sono fondamentali per ottimizzare le prestazioni dell'addestramento parallelo. Sono necessari dei buoni protocolli di partizionamento e comunicazione per ridurre al minimo il sovraccarico di trasferimento.

7.11.3 Confronto

Riassumendo, Tabella 7.5 illustra alcune delle caratteristiche chiave per confrontare il parallelismo dei dati e quello dei modelli.

Tabella 7.5: Confronto tra parallelismo dei dati e parallelismo del modello.

Caratteristica	Parallelismo dei dati	Parallelismo del modello
Definizione	Distribuisce i dati tra i dispositivi con repliche	Distribuisce il modello tra i dispositivi
Obiettivo	Accelera il training tramite il ridimensionamento del calcolo	Abilita un training del modello più ampio
Metodo di Ridimensionamento	Dispositivi/workers in scala	Dimensioni modello in scala
Vincolo Principale	Dimensione del modello per ogni dispositivo	Overhead di coordinamento dispositivo
Requisiti Hardware	Più GPU/TPU	Spesso interconnessione specializzata
Difficoltà Principale	Sincronizzazione dei parametri	Partizionamento e comunicazione complicati
Tipi	N/D	Per livello, per filtro, spaziale
Complessità del Codice	Modifiche minime	Intervento più significativa sul modello
Librerie Popolari	Horovod, PyTorch Distributed	Mesh TensorFlow

7.12 Conclusione

In questo capitolo abbiamo trattato le basi fondamentali che consentono un training efficace dei modelli di intelligenza artificiale. Abbiamo esplorato concetti matematici come funzioni di perdita, backpropagation e discesa del gradiente che rendono possibile l'ottimizzazione delle reti neurali. Abbiamo anche discusso tecniche pratiche per sfruttare i dati di training, la regolarizzazione, la messa a punto degli iperparametri, l'inizializzazione dei pesi e strategie di parallelizzazione distribuita che migliorano convergenza, generalizzazione e scalabilità. Queste metodologie costituiscono il fondamento attraverso cui è stato raggiunto il successo del deep learning nell'ultimo decennio. Padroneggiare questi fondamenti prepara i professionisti a progettare sistemi e perfezionare modelli su misura per il loro contesto. Tuttavia, man mano che modelli e set di dati crescono in modo esponenziale, i sistemi di training devono ottimizzare parametri come tempo, costo e "carbon footprint" [impatto ambientale]. Il ridimensionamento hardware tramite grosse warehouse consente un throughput computazionale enorme, ma le ottimizzazioni relative a efficienza e specializzazione saranno fondamentali. Tecniche software come compressione e sfruttamento delle matrici sparse possono aumentare i guadagni hardware. Ne discuteremo diverse nei prossimi capitoli.

Nel complesso, i fondamenti trattati in questo capitolo preparano i professionisti a costruire, perfezionare e distribuire modelli. Tuttavia, le competenze interdisciplinari che abbracciano teoria, sistemi e hardware differenzieranno gli esperti in grado di portare l'IA al livello successivo in modo sostenibile e responsabile, come richiesto dalla società. Comprendere l'efficienza insieme all'accuratezza costituisce l'approccio ingegneristico bilanciato necessario per addestrare sistemi intelligenti che si integrano senza problemi in molti contesti del mondo reale.

7.13 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Stiamo lavorando costantemente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Thinking About Loss.
- Minimizing Loss.
- Training, Validation, and Test Data.
- Continuous Training:
  - Retraining Trigger.
  - Data Processing Overview.
  - Data Ingestion.
  - Data Validation.
  - Data Transformation.
  - Training with AutoML.
  - Continuous Training with Transfer Learning.
  - Continuous Training Use Case Metrics.
  - Continuous Training Impact on MLOps.

Video

- Video 7.1
- Video 7.2
- Video 7.3
- Video 7.4
- Video 7.5
- Video 7.6
- Video 7.7
- Video 7.8

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 7.1
- Esercizio 7.2
- Esercizio 7.3
- Esercizio 7.5
- Esercizio 7.4

⚠ Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

8. IA Efficiente

DALL·E 3 Prompt: Un'illustrazione concettuale che raffigura l'efficienza nell'intelligenza artificiale usando un'analogia con il cantiere navale. La scena mostra un cantiere navale in fermento dove i container rappresentano bit o byte di dati. Questi container vengono spostati in modo efficiente da gru e veicoli, a simboleggiare l'elaborazione semplificata e rapida delle informazioni nei sistemi di intelligenza artificiale. Il cantiere navale è organizzato meticolosamente, a dimostrazione del concetto di prestazioni ottimali entro i vincoli delle risorse limitate. Sullo sfondo, le navi sono attraccate, a rappresentare diverse piattaforme e scenari in cui viene applicata l'intelligenza artificiale. L'atmosfera dovrebbe trasmettere una tecnologia avanzata con la sostenibilità come tema di fondo e un'ampia applicabilità.

L'efficienza nell'intelligenza artificiale non è semplicemente un lusso, ma una necessità. In questo capitolo, approfondiamo i concetti chiave alla base dell'efficienza dei sistemi di IA. Le richieste computazionali sulle reti neurali possono essere scoraggianti, anche per i sistemi minimali. Per integrare perfettamente l'IA nei dispositivi quotidiani e nei sistemi essenziali, deve funzionare in modo ottimale entro i vincoli delle risorse limitate, mantenendo al contempo la sua efficacia. La ricerca dell'efficienza garantisce che i modelli di IA siano semplificati, rapidi e sostenibili, ampliando così la loro applicabilità su varie piattaforme e scenari.

Obiettivi dell'Apprendimento

- Riconoscere la necessità di un'intelligenza artificiale efficiente nei dispositivi TinyML/edge.
- Comprendere la necessità di architetture di modelli efficienti come MobileNets e SqueezeNet.
- Comprendere perché le tecniche per la compressione dei modelli sono importanti.
- Apprezzare per il valore di un hardware AI efficiente.
- Riconoscere l'importanza delle rappresentazioni numeriche e della loro precisione.
- Comprendere le sfumature del confronto dei modelli oltre la semplice accuratezza.
- Riconoscere che il confronto dei modelli coinvolge memoria, elaborazione, potenza e velocità, non solo accuratezza.
- Riconoscere che l'efficienza comprende tecnologia, costi ed etica.

L'attenzione è rivolta all'acquisizione di una comprensione concettuale delle motivazioni e del significato delle varie strategie per raggiungere un'intelligenza artificiale efficiente, sia in

termini di tecniche che di prospettiva olistica. I capitoli successivi forniscono un'analisi più approfondita di questi molteplici concetti.

8.1 Introduzione

I modelli di training possono consumare molta energia, a volte equivalente all'impatto ambientale di processi industriali considerevoli. Tratteremo alcuni di questi dettagli sulla sostenibilità nel capitolo Sostenibilità dell'IA. Dal punto di vista dell'implementazione, se questi modelli non sono ottimizzati per l'efficienza, possono esaurire rapidamente le batterie dei dispositivi, richiedere una memoria eccessiva o non soddisfare le esigenze di elaborazione in tempo reale. In questo capitolo, miriamo a chiarire le sfumature dell'efficienza, gettando le basi per un'esplorazione completa nei capitoli successivi.

8.2 La Necessità di un'IA Efficiente

L'efficienza assume connotazioni diverse a seconda di dove si verificano i calcoli dell'IA. Rivediamo Cloud, Edge e TinyML (come discusso in Sistemi di ML) e distinguiamoli in termini di efficienza. Figura 8.1 fornisce un confronto generale delle tre diverse piattaforme.

Figura 8.1: Cloud, Mobile e TinyML. Fonte: Schizas et al. (2022).

IA Cloud: I modelli IA tradizionali vengono spesso eseguiti in data center su larga scala dotati di potenti GPU e TPU (Barroso, Hölzle, e Ranganathan 2019). Qui, l'efficienza riguarda l'ottimizzazione delle risorse di calcolo, la riduzione dei costi e la garanzia di elaborazione e restituzione tempestive dei dati. Tuttavia, fare affidamento sul cloud introduce latenza, soprattutto quando si ha a che fare con flussi di dati di grandi dimensioni che richiedono caricamento, elaborazione e download.

IA Edge: L'edge computing avvicina l'intelligenza artificiale alla fonte dei dati, elaborando le informazioni direttamente su dispositivi locali come smartphone, fotocamere o macchine industriali (E. Li et al. 2020). Qui, l'efficienza comprende risposte rapide in tempo reale e ridotte esigenze di trasmissione dei dati. Tuttavia, i vincoli sono più severi: questi dispositivi, sebbene più potenti dei microcontrollori, hanno una potenza di calcolo limitata rispetto alle configurazioni cloud.

TinyML: TinyML supera i limiti consentendo ai modelli di intelligenza artificiale di funzionare su microcontrollori o ambienti con risorse estremamente limitate. La differenza di prestazioni del processore e della memoria tra TinyML e i sistemi cloud o mobili può essere di diversi ordini di grandezza (Warden e Situnayake 2019). L'efficienza in TinyML consiste nell'assicurare che i modelli siano sufficientemente leggeri da adattarsi a questi dispositivi,

consumino il minimo di energia (fondamentale per i dispositivi alimentati a batteria) e continuino a svolgere le loro attività in modo efficace.
Lo spettro da Cloud a TinyML rappresenta un passaggio da vaste risorse di elaborazione centralizzate ad ambienti distribuiti, localizzati e limitati. Passando dall'uno all'altro, i problemi e le strategie relative all'efficienza evolvono, sottolineando la necessità di approcci specializzati su misura per ogni scenario. Dopo aver stabilito la necessità di un'intelligenza artificiale efficiente, in particolare nel contesto di TinyML, passeremo all'esplorazione delle metodologie ideate per rispondere a queste sfide. Le sezioni seguenti delineano i concetti principali che approfondiremo in seguito. Dimostreremo l'ampiezza e la profondità dell'innovazione necessarie per ottenere un'intelligenza artificiale efficiente mentre esploriamo queste strategie.

8.3 Architetture di Modelli Efficienti

Selezionare un'architettura del modello ottimale è tanto cruciale quanto ottimizzarla. Negli ultimi anni, i ricercatori hanno compiuto passi da gigante nell'esplorazione di architetture innovative che possono avere intrinsecamente meno parametri pur mantenendo prestazioni elevate.

MobileNet: MobileNet sono modelli di applicazioni di visione mobile ed embedded efficienti (Howard et al. 2017). L'idea chiave che ha portato al loro successo sono le convoluzioni separabili in profondità, che riducono significativamente il numero di parametri e calcoli nella rete. MobileNetV2 e V3 migliorano ulteriormente questo design introducendo residui invertiti e colli di bottiglia lineari.

SqueezeNet: SqueezeNet è una classe di modelli ML noti per le sue dimensioni ridotte senza sacrificare la precisione. Ciò si ottiene utilizzando un "modulo fire" che riduce il numero di canali di input a filtri 3x3, riducendo così i parametri (Iandola et al. 2016). Inoltre, impiega il downsampling [sottocampionamento] ritardato per aumentare la precisione mantenendo una mappa delle feature più ampia.

Varianti ResNet: L'architettura Residual Network (ResNet) consente l'introduzione di connessioni skip o scorciatoie (K. He et al. 2016). Alcune varianti di ResNet sono progettate per essere più efficienti. Ad esempio, ResNet-SE incorpora il meccanismo "squeeze and excitation" per ricalibrare le feature map (J. Hu, Shen, e Sun 2018), mentre ResNeXt offre convoluzioni raggruppate per l'efficienza (S. Xie et al. 2017).

8.4 Compressione Efficiente del Modello

I metodi di compressione dei modelli sono essenziali per portare i modelli di apprendimento profondo su dispositivi con risorse limitate. Queste tecniche riducono le dimensioni dei modelli, il consumo energetico e le richieste di elaborazione senza perdere significativamente la precisione. Ad alto livello, i metodi possono essere categorizzati nei seguenti metodi fondamentali:

Pruning: L'Abbiamo menzionato un paio di volte nei capitoli precedenti, ma non l'abbiamo ancora formalmente introdotta. Il pruning è simile alla potatura dei rami di un albero. Questo è stato pensato per la prima volta nel documento Optimal Brain Damage (LeCun, Denker, e Solla 1989) ed è stato successivamente reso popolare nel contesto del deep learning da Han, Mao, e Dally (2016). Determinati pesi o interi neuroni vengono rimossi dalla rete nella potatura in base a criteri specifici. Questo può ridurre significativamente le dimensioni del modello. In Sezione 9.2.1 esploreremo due delle principali strategie di potatura, quella strutturata e quella non-strutturata. Figura 8.2 è un esempio di potatura della rete neurale, in

cui la rimozione di alcuni nodi negli strati interni (in base a criteri specifici) riduce il numero di rami tra i nodi e, a sua volta, le dimensioni del modello.

Figura 8.2: Neural Network Pruning.

Quantizzazione: La quantizzazione è il processo di limitazione di un input da un set ampio a un output in un set più piccolo, principalmente nel deep learning; ciò significa ridurre il numero di bit che rappresentano i pesi e i bias del modello. Ad esempio, l'utilizzo di rappresentazioni a 16 o 8 bit anziché a 32 bit può ridurre la dimensione del modello e velocizzare i calcoli, con un piccolo compromesso in termini di accuratezza. Esploreremo questi aspetti più in dettaglio in Sezione 9.3.4. Figura 8.3 mostra un esempio di quantizzazione mediante arrotondamento al numero più vicino. La conversione da virgola mobile a 32 bit a 16 bit riduce l'utilizzo della memoria del 50%. Passare da un intero a 32 bit a uno a 8 bit riduce l'utilizzo della memoria del 75%. Mentre la perdita di precisione numerica e, di conseguenza, di prestazioni del modello è minima, l'efficienza nell'utilizzo della memoria è significativa.

Figura 8.3: Diverse forme di quantizzazione.

Knowledge Distillation: La "distillazione della conoscenza" comporta l'addestramento di un modello più piccolo (studente) per replicare il comportamento di un modello più grande (insegnante). L'idea è quella di trasferire la conoscenza dal modello ingombrante a quello leggero. Quindi, il modello più piccolo raggiunge prestazioni vicine alla sua controparte più grande ma con parametri significativamente inferiori. Figura 8.4 mostra la struttura tutor-studente per la distillazione della conoscenza. Esploreremo la "distillazione della conoscenza" in modo più dettagliato in Sezione 9.2.2.1.

Figura 8.4: Il framework tutor-studente per la distillazione della conoscenza. Fonte: Medium

8.5 Hardware di Inferenza Efficiente

Nel capitolo Training, abbiamo discusso il processo di training dei modelli di intelligenza artificiale. Ora, dal punto di vista dell'efficienza, è importante notare che il training è un'attività che richiede molte risorse e molto tempo, spesso richiede hardware potente e impiega da ore a settimane per essere completato. L'inferenza, d'altra parte, deve essere il più veloce possibile, soprattutto nelle applicazioni in tempo reale. È qui che entra in gioco un hardware di inferenza efficiente. Ottimizzando l'hardware specificamente per le attività di inferenza, possiamo ottenere tempi di risposta rapidi e un funzionamento efficiente dal punto di vista energetico, il che è particolarmente cruciale per i dispositivi edge e i sistemi embedded.

TPU (Tensor Processing Unit): Le TPU sono ASIC (Application-Specific Integrated Circuits) personalizzati da Google per accelerare i carichi di lavoro di apprendimento automatico (N. P. Jouppi et al. 2017a). Sono ottimizzate per le operazioni tensoriali, offrono un throughput elevato per l'aritmetica a bassa precisione e sono progettate specificamente per il machine learning delle reti neurali. Le TPU accelerano significativamente l'addestramento e l'inferenza del modello rispetto alle GPU/CPU generiche. Questo potenziamento si traduce in un addestramento più rapido dei modelli e in capacità di inferenza in tempo reale o quasi reale, fondamentali per applicazioni come la ricerca vocale e la realtà aumentata.
Le Edge TPU sono una versione più piccola e a basso consumo delle TPU di Google, studiate appositamente per i dispositivi edge. Forniscono un'inferenza ML veloce sul dispositivo per i modelli TensorFlow Lite. Le Edge TPU consentono un'inferenza a bassa latenza e ad alta efficienza su dispositivi edge come smartphone, dispositivi IoT e sistemi embedded. Le capacità di IA possono essere implementate in applicazioni in tempo reale senza comunicare con un server centrale, risparmiando così larghezza di banda e riducendo la latenza. Si consideri la tabella in Figura 8.5. Mostra le differenze di prestazioni tra l'esecuzione di modelli diversi su CPU rispetto a un acceleratore Coral USB. L'acceleratore Coral USB è un

accessorio della piattaforma Coral AI di Google che consente agli sviluppatori di collegare le Edge TPU ai computer Linux. L'esecuzione dell'inferenza sulle Edge TPU è stata da 70 a 100 volte più veloce rispetto alle CPU.

Model architecture	Desktop CPU*	Desktop CPU* + USB Accelerator (USB 3.0) *with Edge TPU*	Embedded CPU **	Dev Board † *with Edge TPU*
MobileNet v1	47 ms	2.2 ms	179 ms	2.2 ms
MobileNet v2	45 ms	2.3 ms	150 ms	2.5 ms
Inception v1	92 ms	3.6 ms	406 ms	3.9 ms
Inception v4	792 ms	100 ms	3,463 ms	100 ms

Figura 8.5: Confronto delle prestazioni tra acceleratore e CPU in diverse configurazioni hardware. Desktop CPU: 64-bit Intel(R) Xeon(R) E5-1650 v4 @ 3.60GHz. Embedded CPU: Quad-core Cortex-A53 @ 1.5GHz, †Dev Board: Quad-core Cortex-A53 @ 1.5GHz + Edge TPU. Fonte: TensorFlow Blog.

Acceleratori NN (Neural Network): Gli acceleratori di reti neurali a funzione fissa sono acceleratori hardware progettati esplicitamente per i calcoli di reti neurali. Possono essere chip standalone o far parte di una soluzione di system-on-chip (SoC) più ampia. Ottimizzando l'hardware per le operazioni specifiche richieste dalle reti neurali, come moltiplicazioni di matrici e convoluzioni, gli acceleratori NN possono ottenere tempi di inferenza più rapidi e consumi energetici inferiori rispetto alle CPU e alle GPU per uso generico. Sono particolarmente utili nei dispositivi TinyML con vincoli di potenza o termici, come smartwatch, micro-droni o robotica.

Ma questi sono solo gli esempi più comuni. Stanno emergendo diversi altri tipi di hardware che hanno il potenziale per offrire vantaggi significativi per l'inferenza. Questi includono, ma non solo, hardware neuromorfico, elaborazione fotonica, ecc. In Sezione 10.3, esploreremo questi aspetti in modo più dettagliato.

Un hardware efficiente per l'inferenza velocizza il processo, risparmia energia, prolunga la durata della batteria e può funzionare in condizioni di tempo reale. Man mano che l'intelligenza artificiale viene integrata in innumerevoli applicazioni, dalle telecamere intelligenti agli assistenti vocali, il ruolo dell'hardware ottimizzato diventerà sempre più importante. Sfruttando questi componenti hardware specializzati, sviluppatori e ingegneri possono portare la potenza dell'intelligenza artificiale a dispositivi e situazioni che prima erano impensabili.

8.6 Matematica Efficiente

L'apprendimento automatico, e in particolare il deep learning, comporta enormi quantità di elaborazione. I modelli possono avere milioni o miliardi di parametri, spesso addestrati su vasti set di dati. Ogni operazione, ogni moltiplicazione o addizione, richiede risorse di

elaborazione. Pertanto, la precisione dei numeri utilizzati in queste operazioni può avere un impatto significativo sulla velocità di elaborazione, sul consumo di energia e sui requisiti di memoria. È qui che entra in gioco il concetto di numeri efficienti.

8.6.1 Formati Numerici

Esistono molti tipi diversi di numeri. I numeri hanno una lunga storia nei sistemi di elaborazione.

Floating point: Noto come "virgola mobile" a precisione singola, FP32 utilizza 32 bit per rappresentare un numero, incorporandone segno, esponente e mantissa. Comprendere come i numeri in virgola mobile sono rappresentati in modo approfondito è fondamentale per comprendere le varie ottimizzazioni possibili nei calcoli numerici. Il bit del segno determina se il numero è positivo o negativo, l'esponente controlla l'intervallo di valori che possono essere rappresentati e la mantissa determina la precisione del numero. La combinazione di questi componenti consente ai numeri in virgola mobile di rappresentare un'ampia gamma di valori con vari gradi di precisione.
Video 8.1 fornisce una panoramica completa di questi tre componenti principali, segno, esponente e mantissa, e di come funzionano insieme per rappresentare i numeri in virgola mobile.

Video 8.1: Numeri in Virgola Mobile

https://youtu.be/gc1Nl3mmCuY?si=nImcymfbE5H392vu

FP32 è ampiamente adottato in molti framework di deep learning e bilancia accuratezza e requisiti computazionali. È prevalente nella fase di training per molte reti neurali grazie alla sua sufficiente precisione nel catturare dettagli minuti durante gli aggiornamenti dei pesi.
Noto anche come virgola mobile a mezza precisione, FP16 utilizza 16 bit per rappresentare un numero, inclusi il segno, l'esponente e la frazione. Offre un buon equilibrio tra precisione e risparmio di memoria. FP16 è particolarmente popolare nella training di deep learning su GPU che supportano l'aritmetica a precisione mista, combinando i vantaggi di velocità di FP16 con la precisione di FP32 quando necessario.
Figura 8.6 mostra tre diversi formati in virgola mobile: Float32, Float16 e BFloat16.

Figura 8.6: Tre formati a virgola mobile.

Diversi altri formati numerici rientrano in una classe esotica. Un esempio esotico è BF16 o Brain Floating Point. È un formato numerico a 16 bit progettato esplicitamente per applicazioni di deep learning. È un compromesso tra FP32 e FP16, che mantiene l'esponente a 8 bit di FP32 riducendo la mantissa a 7 bit (rispetto alla mantissa a 23 bit di FP32). Questa struttura dà priorità al range rispetto alla precisione. BF16 ha ottenuto risultati di training paragonabili in accuratezza a FP32, utilizzando significativamente meno memoria e risorse computazionali (Kalamkar et al. 2019). Ciò lo rende adatto non solo per l'inferenza, ma anche per il training di reti neurali profonde.
Mantenendo l'esponente a 8 bit di FP32, BF16 offre un range simile, che è fondamentale per le attività di deep learning in cui determinate operazioni possono generare numeri molto grandi o molto piccoli. Allo stesso tempo, troncando la precisione, BF16 consente requisiti di memoria e computazionali ridotti rispetto a FP32. BF16 è emerso come una promettente via di mezzo nel panorama dei formati numerici per il deep learning, fornendo un'alternativa efficiente ed efficace ai formati FP32 e FP16 più tradizionali.

Intero: Si tratta di rappresentazioni di numeri interi che utilizzano 8, 4 e 2 bit. Vengono spesso utilizzati durante la fase di inferenza delle reti neurali, in cui i pesi e le attivazioni del modello sono quantizzati a queste precisioni inferiori. Le rappresentazioni intere sono deterministiche e offrono notevoli vantaggi in termini di velocità e memoria rispetto alle rappresentazioni in virgola mobile. Per molte attività di inferenza, in particolare sui dispositivi edge, la leggera perdita di accuratezza dovuta alla quantizzazione è spesso accettabile, dati i guadagni di efficienza. Una forma estrema di numeri interi è per le reti neurali binarie (BNN), in cui pesi e attivazioni sono vincolati a uno dei due valori: +1 o -1.

Larghezze di bit variabili: Oltre alle larghezze standard, sono in corso ricerche su numeri con larghezze di bit estremamente basse, persino fino a rappresentazioni binarie o ternarie. Le operazioni con larghezze di bit estremamente basse possono offrire accelerazioni significative e ridurre ulteriormente il consumo di energia. Sebbene permangano dei problemi nel mantenere l'accuratezza del modello con una quantizzazione così drastica, si continuano a fare progressi in quest'area.
L'efficienza numerica non riguarda solo la riduzione della larghezza di bit dei numeri, ma anche la comprensione dei compromessi tra accuratezza ed efficienza. Man mano che i modelli di apprendimento automatico diventano più pervasivi, soprattutto in ambienti reali con risorse limitate, l'attenzione su una numerica efficiente continuerà a crescere. Selezionando e sfruttando attentamente la precisione numerica appropriata, è possibile ottenere prestazioni di modello robuste ottimizzando al contempo velocità, memoria ed energia. Tabella 8.1 riassume questi compromessi.

Tabella 8.1: Confronto dei livelli di precisione nel deep learning.

Precisione	Pro	Contro
FP32 (virgola mobile a 32 bit)	• Precisione standard utilizzata nella maggior parte dei framework di deep learning. • Elevata accuratezza grazie all'ampia capacità di rappresentazione. • Adatto per il training	• Elevato utilizzo di memoria. • Tempi di inferenza più lenti rispetto ai modelli quantizzati. • Maggiore consumo energetico.
FP16 (virgola mobile a 16 bit)	• Riduce l'utilizzo di memoria rispetto a FP32. • Velocizza i calcoli su hardware che supporta FP16.	• Minore capacità di rappresentazione rispetto a FP32. • Rischio di instabilità numerica in alcuni modelli o livelli.

	• Spesso utilizzato nel training a precisione mista per bilanciare velocità e accuratezza.	
INT8 (intero a 8 bit)	• Impronta di memoria notevolmente ridotta rispetto alle rappresentazioni in virgola mobile. • Inferenza più rapida se l'hardware supporta i calcoli INT8. • Adatto a molti scenari di quantizzazione post-training.	• La quantizzazione può comportare una certa perdita di accuratezza. • Richiede una calibrazione attenta durante la quantizzazione per ridurre al minimo il degrado della precisione.
INT4 (intero a 4 bit)	• Utilizzo di memoria ancora inferiore rispetto a INT8. • Ulteriore potenziale di accelerazione per l'inferenza.	• Rischio di perdita di precisione più elevato rispetto a INT8. • La calibrazione durante la quantizzazione diventa più critica.
Binario	• Ingombro di memoria minimo (solo 1 bit per parametro). • Inferenza estremamente rapida grazie alle operazioni bit a bit. • Efficienza energetica.	• Calo significativo della precisione per molte attività. • Dinamiche di training complesse grazie alla quantizzazione estrema.
Ternario	• Basso utilizzo di memoria ma leggermente superiore a quello binario. • Offre una via di mezzo tra rappresentazione ed efficienza.	• L'accuratezza potrebbe essere ancora inferiore a quella dei modelli di precisione più elevata. • Le dinamiche di addestramento possono essere complesse.

8.6.2 Vantaggi dell'Efficienza

L'efficienza numerica è importante per i carichi di lavoro di machine learning per diversi motivi:

Efficienza Computazionale: I calcoli ad alta precisione (come FP32 o FP64) possono essere lenti e richiedere molte risorse. Ridurre la precisione numerica può ottenere tempi di calcolo più rapidi, specialmente su hardware specializzato che supporta una precisione inferiore.

Efficienza della Memoria: I requisiti di archiviazione diminuiscono con una precisione numerica ridotta. Ad esempio, FP16 richiede metà della memoria di FP32. Ciò è fondamentale quando si distribuiscono modelli su dispositivi edge con memoria limitata o si lavora con modelli di grandi dimensioni.

Efficienza Energetica: I calcoli a precisione inferiore spesso consumano meno energia, il che è particolarmente importante per i dispositivi alimentati a batteria.

Introduzione del Rumore: È interessante notare che il rumore introdotto utilizzando una precisione inferiore può talvolta fungere da regolarizzatore, contribuendo a prevenire l'overfitting in alcuni modelli.

Accelerazione Hardware: Molti acceleratori di IA e GPU moderni sono ottimizzati per operazioni di precisione inferiore, sfruttando i vantaggi dell'efficienza di tali numeri.

8.7 Valutazione dei Modelli

Vale la pena notare che i vantaggi e i compromessi effettivi possono variare in base all'architettura specifica della rete neurale, al set di dati, all'attività e all'hardware utilizzato. Prima di decidere una precisione numerica, è consigliabile eseguire esperimenti per valutare l'impatto sull'applicazione desiderata.

8.7.1 Metriche di Efficienza

Una profonda comprensione dei metodi di valutazione dei modelli è importante per guidare questo processo in modo sistematico. Quando si valuta l'efficacia e l'idoneità dei modelli di intelligenza artificiale per varie applicazioni, le metriche di efficienza vengono in primo piano.

I **FLOP (Floating Point Operations)**, introdotti in Training, misurano le esigenze computazionali di un modello. Ad esempio, una moderna rete neurale come BERT ha miliardi di FLOP, che potrebbero essere gestibili su un potente server cloud ma sarebbero gravosi su uno smartphone. FLOP più elevati possono portare a tempi di inferenza più prolungati e a un notevole consumo di energia, soprattutto su dispositivi senza acceleratori hardware specializzati. Quindi, per applicazioni in tempo reale come lo streaming video o i giochi, potrebbero essere più desiderabili modelli con FLOP più bassi.

L'**Utilizzo della Memoria** riguarda la quantità di spazio di archiviazione richiesta dal modello, che influisce sia sullo spazio di archiviazione del dispositivo che sulla RAM. Si prenda in considerazione l'implementazione di un modello su uno smartphone: un modello che occupa diversi gigabyte di spazio non solo consuma prezioso spazio di archiviazione, ma potrebbe anche essere più lento a causa della necessità di caricare grandi pesi nella memoria. Ciò diventa particolarmente cruciale per dispositivi edge come telecamere di sicurezza o droni, dove impronte di memoria minime sono vitali per l'archiviazione e l'elaborazione rapida dei dati.

Il **Consumo Energetico** diventa particolarmente cruciale per i dispositivi che si basano sulle batterie. Ad esempio, un monitor sanitario indossabile che utilizza un modello ad alto consumo energetico potrebbe esaurire la batteria in poche ore, rendendolo poco pratico per il monitoraggio continuo. L'ottimizzazione dei modelli per un basso consumo energetico diventa essenziale mentre ci muoviamo verso un'era dominata dai dispositivi IoT, dove molti dispositivi funzionano a batteria.

Il **Tempo di Inferenza** riguarda la rapidità con cui un modello può produrre risultati. In applicazioni come la guida autonoma, dove decisioni in frazioni di secondo fanno la differenza tra sicurezza e calamità, i modelli devono funzionare rapidamente. Se il modello di un'auto a guida autonoma impiega anche solo pochi secondi in più per riconoscere un ostacolo, le conseguenze potrebbero essere disastrose. Quindi, garantire che il tempo di inferenza di un modello sia allineato con le richieste in tempo reale della sua applicazione è fondamentale.
In sostanza, queste metriche di efficienza sono più che dei numeri che stabiliscono dove e come un modello può essere distribuito in modo efficace. Un modello potrebbe vantare un'elevata accuratezza, ma se i suoi FLOP, l'utilizzo della memoria, il consumo energetico o il tempo di inferenza lo rendono inadatto alla piattaforma prevista o agli scenari del mondo reale, la sua utilità pratica diventa limitata.

8.7.2 Confronti di Efficienza

Il panorama dei modelli di machine learning è vasto, con ogni modello che offre un set unico di punti di forza e considerazioni di implementazione. Sebbene le cifre di accuratezza grezza o le velocità di training e inferenza possano essere parametri di riferimento allettanti, forniscono un quadro incompleto. Un'analisi comparativa più approfondita rivela diversi fattori critici che influenzano l'idoneità di un modello per le applicazioni TinyML. Spesso, incontriamo il delicato equilibrio tra accuratezza ed efficienza. Ad esempio, mentre un modello di deep learning e denso e una variante MobileNet leggera potrebbero eccellere nella classificazione delle immagini, le loro richieste di calcolo potrebbero essere ad estremi opposti. Questa differenziazione è particolarmente pronunciata quando si confrontano le

distribuzioni su server cloud con risorse abbondanti rispetto ai limitati dispositivi TinyML. In molti scenari del mondo reale, i guadagni marginali in termini di accuratezza potrebbero essere oscurati dalle inefficienze di un modello ad alta intensità di risorse richieste.
Inoltre, la scelta del modello ottimale non è sempre universale, ma spesso dipende dalle specifiche di un'applicazione. Ad esempio, un modello che eccelle in scenari di rilevamento di oggetti generali potrebbe avere difficoltà in ambienti di nicchia, come il rilevamento di difetti di fabbricazione in una fabbrica. Questa adattabilità, o la sua mancanza, può influenzare l'utilità reale di un modello.
Un'altra considerazione importante è la relazione tra la complessità del modello e i suoi vantaggi pratici. Prendiamo gli assistenti attivati tramite comando vocale, come "Alexa" o "OK Google". Mentre un modello complesso potrebbe dimostrare una comprensione marginalmente superiore del parlato dell'utente se è più lento a rispondere rispetto a una controparte più semplice, l'esperienza utente potrebbe essere compromessa. Pertanto, l'aggiunta di layer o parametri solo a volte equivale a risultati migliori nel mondo reale.
Un'altra considerazione importante è la relazione tra la complessità del modello e i suoi vantaggi pratici. Prendiamo gli assistenti vocali come "Alexa" o "OK Google". Mentre un modello complesso potrebbe dimostrare una comprensione leggermente superiore del parlato dell'utente se è più lento a rispondere rispetto a una controparte più semplice, l'esperienza utente potrebbe essere compromessa. Pertanto, l'aggiunta di layer o parametri solo a volte equivale a risultati migliori nel mondo reale.
Inoltre, mentre i set di dati di riferimento, come ImageNet (Russakovsky et al. 2015), COCO (T.-Y. Lin et al. 2014), Visual Wake Words (L. Wang e Zhan 2019a), Google Speech Commands (Warden 2018), ecc. forniscono una metrica di prestazioni standardizzata, potrebbero non catturare la diversità e l'imprevedibilità dei dati del mondo reale. Due modelli di riconoscimento facciale con punteggi di riferimento simili potrebbero mostrare competenze diverse quando si trovano di fronte a background etnici diversi o condizioni di illuminazione difficili. Tali disparità sottolineano l'importanza di robustezza e coerenza tra dati diversi. Ad esempio, Figura 8.7 dal set di dati Dollar Street mostra immagini di stufe su redditi mensili estremi. Le stufe hanno forme e livelli tecnologici diversi in diverse regioni e livelli di reddito. Un modello che non è addestrato su set di dati diversi potrebbe funzionare bene su un benchmark ma fallire nelle applicazioni del mondo reale. Quindi, se un modello fosse addestrato solo su immagini di stufe trovate nei paesi ricchi, non riuscirebbe a riconoscere le stufe delle regioni più povere.

Figura 8.7: Diversi tipi di stufe. Fonte: Immagini di stufe di Dollar Street.

In sostanza, un'analisi comparativa approfondita trascende le metriche numeriche. È una valutazione olistica intrecciata con applicazioni del mondo reale, costi e le intricate sottigliezze che ogni modello porta con sé. Ecco perché avere parametri di riferimento e metriche standard ampiamente stabiliti e adottati dalla comunità diventa importante.

8.8 Conclusione

L'intelligenza artificiale efficiente è fondamentale mentre ci spingiamo verso un'implementazione più ampia e diversificata del machine learning nel mondo reale. Questo capitolo ha fornito una panoramica, esplorando le varie metodologie e considerazioni alla base del raggiungimento di un'intelligenza artificiale efficiente, a partire dall'esigenza fondamentale, dalle somiglianze e dalle differenze tra i sistemi cloud, Edge e TinyML. Abbiamo esaminato le architetture dei modelli efficienti e la loro utilità per l'ottimizzazione. Le tecniche di compressione dei modelli come pruning, quantizzazione e distillazione della conoscenza esistono per aiutare a ridurre le richieste di calcolo e l'ingombro della memoria senza influire in modo significativo sulla precisione. Hardware specializzati come TPU e acceleratori NN offrono chip ottimizzati per le operazioni di rete neurale e il flusso di dati. I numeri efficienti bilanciano precisione ed efficienza, consentendo ai modelli di ottenere prestazioni robuste utilizzando risorse minime. Esploreremo questi argomenti in modo approfondito e dettagliato nei capitoli successivi.

Insieme, questi formano un quadro olistico per un'intelligenza artificiale efficiente. Ma il viaggio non finisce qui. Il raggiungimento di un'intelligenza efficiente in modo ottimale richiede ricerca e innovazione continue. Man mano che i modelli diventano più sofisticati, i set di dati crescono e le applicazioni si diversificano in domini specializzati, l'efficienza deve evolversi di pari passo. La misura dell'impatto nel mondo reale richiede parametri di riferimento adatti e metriche standardizzate che vadano oltre le semplicistiche cifre dell'accuratezza.

Inoltre, l'intelligenza artificiale efficiente si espande oltre l'ottimizzazione tecnologica e comprende costi, impatto ambientale e considerazioni etiche per il bene della società in

senso più ampio. Man mano che l'intelligenza artificiale permea i settori e la vita quotidiana, una prospettiva completa sull'efficienza sostiene il suo progresso sostenibile e responsabile. I capitoli successivi si baseranno su questi concetti fondamentali, fornendo approfondimenti concreti e norme pratiche per lo sviluppo e l'implementazione di soluzioni di intelligenza artificiale efficienti.

8.9 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Deploying on Edge Devices: challenges and techniques.
- Model Evaluation.
- Continuous Evaluation Challenges for TinyML.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- *Prossimamente*.

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

9. Ottimizzazioni dei Modelli

DALL·E 3 Prompt: Illustrazione di un modello di rete neurale rappresentato come un cantiere edile affollato, con un gruppo eterogeneo di operai edili, sia uomini che donne, di varie etnie, etichettati come 'pruning', 'quantization' e 'sparsity'. Stanno lavorando insieme per rendere la rete neurale più efficiente e più piccola, mantenendo un'elevata precisione. L'operaio che si occupa del 'pruning', una donna ispanica, sta tagliando le connessioni non necessarie dal centro della rete. L'addetto alla 'quantization', un uomo caucasico, sta regolando o modificando i pesi ovunque. L'operaio che si occupa di 'sparsity', una donna africana, sta rimuovendo i nodi non necessari per ridurre il modello. Sullo sfondo, camion e gru edili assistono gli operai nei loro compiti. La rete neurale si sta trasformando visivamente da una struttura complessa e grande a una più snella e piccola.

Quando i modelli di apprendimento automatico vengono distribuiti su sistemi, in particolare su sistemi embedded con risorse limitate, l'ottimizzazione dei modelli è una necessità. Mentre il machine learning richiede spesso risorse computazionali sostanziali, i sistemi sono intrinsecamente limitati in termini di memoria, potenza di elaborazione ed energia. Questo capitolo si immergerà nell'arte e nella scienza dell'ottimizzazione dei modelli di machine learning per garantire che siano leggeri, efficienti ed efficaci quando distribuiti in scenari TinyML.

Obiettivi dell'Apprendimento

- Apprendere tecniche come "pruning", "knowledge distillation" e architetture di modelli specializzate per rappresentare i modelli in modo più efficiente
- Comprendere i metodi di quantizzazione per ridurre le dimensioni del modello e consentire un'inferenza più rapida tramite numeri con precisione ridotta
- Esplorare approcci di ottimizzazione basati sull'hardware per abbinare i modelli alle capacità del dispositivo target
- Sviluppare un pensiero olistico per bilanciare i compromessi in termini di complessità del modello, accuratezza, latenza, potenza ecc. in base ai requisiti dell'applicazione
- Scoprire strumenti software come framework e piattaforme di conversione del modello che consentono l'implementazione di modelli ottimizzati
- Ottenere informazioni strategiche sulla selezione e l'applicazione di ottimizzazioni del modello in base ai vincoli del caso d'uso e ai target hardware

9.1 Introduzione

L'ottimizzazione dei modelli di apprendimento automatico per l'implementazione pratica è un aspetto critico dei sistemi di IA. Questo capitolo si concentra sull'esplorazione delle tecniche di ottimizzazione dei modelli in relazione allo sviluppo di sistemi di apprendimento automatico, che vanno dalle considerazioni di architettura del modello di alto livello agli adattamenti hardware di basso livello. Figura 9.1 Illustra i tre livelli dello stack di ottimizzazione che trattiamo.

Figura 9.1: Tre livelli da coprire.

Al livello più alto, esaminiamo le metodologie per ridurre la complessità dei parametri del modello senza compromettere le capacità inferenziali. Tecniche come la potatura e la distillazione della conoscenza offrono approcci potenti per comprimere e perfezionare i modelli mantenendo o addirittura migliorandone le prestazioni, non solo in termini di qualità del modello ma anche nelle prestazioni effettive del runtime del sistema. Questi metodi sono fondamentali per creare modelli efficienti che possono essere implementati in ambienti con risorse limitate.

Inoltre, esploriamo il ruolo della precisione numerica nei calcoli del modello. Comprendere come diversi livelli di precisione numerica influenzino le dimensioni, la velocità e l'accuratezza del modello è essenziale per ottimizzare le prestazioni. Analizziamo vari formati numerici e l'applicazione dell'aritmetica a precisione ridotta, particolarmente rilevante per le distribuzioni di sistemi embedded in cui le risorse computazionali sono spesso limitate.

Al livello più basso, esploriamo l'intricato panorama della progettazione congiunta hardware-software. Questa esplorazione rivela come i modelli possono essere personalizzati per sfruttare le caratteristiche e le capacità specifiche delle piattaforme hardware di destinazione. Allineando la progettazione del modello all'architettura hardware, possiamo migliorare significativamente le prestazioni e l'efficienza.

Questo approccio collettivo si concentra sull'aiutarci a sviluppare e distribuire modelli di apprendimento automatico efficienti, potenti e consapevoli dell'hardware. Dalla semplificazione delle architetture del modello alla messa a punto della precisione numerica e all'adattamento a hardware specifico, questo capitolo copre l'intero spettro di strategie di ottimizzazione. Alla conclusione di questo capitolo, i lettori avranno acquisito una conoscenza approfondita di varie tecniche di ottimizzazione e delle loro applicazioni pratiche in scenari del mondo reale. Questa conoscenza è importante per creare modelli di apprendimento automatico che non solo funzionino bene, ma siano anche ottimizzati per i vincoli e le opportunità offerti dagli ambienti informatici moderni.

9.2 Rappresentazione Efficiente del Modello

Il primo passo per l'ottimizzazione del modello inizia in un territorio familiare per la maggior parte dei professionisti del ML: la rappresentazione efficiente del modello viene spesso affrontata per la prima volta al livello più alto di astrazione della parametrizzazione, ovvero l'architettura stessa del modello.

La maggior parte dei professionisti del ML tradizionali progetta modelli con un obiettivo generale di alto livello in mente, che si tratti di classificazione delle immagini, rilevamento di persone o individuazione di parole chiave come menzionato in precedenza in questo testo. I loro progetti in genere finiscono per adattarsi naturalmente ad alcuni vincoli soft dovuti a risorse di elaborazione limitate durante lo sviluppo, ma in genere questi progetti non sono a conoscenza di vincoli successivi, come quelli richiesti se il modello deve essere distribuito su un dispositivo più limitato anziché sul cloud.

In questa sezione, discuteremo di come i professionisti possono sfruttare i principi della progettazione congiunta hardware-software anche nell'architettura di alto livello di un modello per rendere i loro modelli compatibili con i dispositivi edge. Da quelli più consapevoli dell'hardware a quelli meno consapevoli a questo livello di modifica, discutiamo alcune delle strategie più comuni per una parametrizzazione efficiente del modello: pruning, compressione e architetture edge-friendly. Abbiamo già parlato di pruning e compressione del modello in Sezione 8.4; questa sezione andrà oltre le definizioni per fornire una comprensione tecnica del loro funzionamento.

9.2.1 Il Pruning

9.2.1.1 Panoramica

Il model pruning [potatura] è una tecnica di apprendimento automatico che riduce le dimensioni e la complessità di un modello di rete neurale, mantenendone il più possibile le capacità predittive. L'obiettivo della potatura è quello di rimuovere componenti ridondanti o non essenziali del modello, tra cui connessioni tra neuroni, singoli neuroni o persino interi layer della rete.

Questo processo in genere comporta l'analisi del modello di machine learning per identificare e rimuovere pesi, nodi o layer che hanno scarso impatto sugli output del modello. Potando selettivamente un modello in questo modo, il numero totale di parametri può essere ridotto in modo significativo senza cali sostanziali nell'accuratezza del modello. Il modello compresso risultante richiede meno memoria e risorse di calcolo per l'addestramento e l'esecuzione, consentendo tempi di inferenza più rapidi.

Il pruning del modello è particolarmente utile quando si distribuiscono modelli di apprendimento automatico su dispositivi con risorse di calcolo limitate, come telefoni cellulari o sistemi TinyML. La tecnica facilita la distribuzione di modelli più grandi e complessi su questi dispositivi riducendo le loro richieste di risorse. Inoltre, i modelli più piccoli richiedono meno dati per generalizzare bene e sono meno inclini all'overfitting [sovradattamento]. Fornendo un modo efficiente per semplificare i modelli, la potatura dei modelli è diventata una tecnica fondamentale per ottimizzare le reti neurali nell'apprendimento automatico.

Esistono diverse tecniche di potatura comuni utilizzate nell'apprendimento automatico, tra cui la potatura strutturata, la potatura non strutturata, la potatura iterativa, la potatura bayesiana e persino la potatura casuale. Oltre a potare i pesi, si possono anche potare le attivazioni. La potatura di attivazioni prende di mira specificamente neuroni o filtri che si attivano raramente o hanno un'attivazione complessivamente bassa. Esistono numerosi altri metodi, come la potatura di sensibilità e movimento. Per un elenco completo dei metodi, si

consiglia al lettore di leggere il seguente articolo: "A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations" (2023).
Quindi, come si scelgono i metodi di potatura? Esistono molte varianti di tecniche di potatura, ciascuna delle quali varia l'euristica di ciò che dovrebbe essere mantenuto e potato dal modello, nonché il numero di volte in cui cui deve essere eseguita. Tradizionalmente, la potatura avviene dopo che il modello è completamente addestrato, dove il modello potato può subire una lieve perdita di accuratezza. Tuttavia, come discuteremo più avanti, recenti scoperte hanno trovato che la potatura può essere utilizzata durante l'addestramento (ad esempio, in modo iterativo) per identificare rappresentazioni del modello più efficienti e accurate.

9.2.1.2 Potatura Strutturata

Iniziamo con la "potatura strutturata", una tecnica che riduce le dimensioni di una rete neurale eliminando intere sotto-strutture specifiche del modello mantenendone la struttura generale. Rimuove interi neuroni/canali o layer in base a criteri di importanza. Ad esempio, per una rete neurale convoluzionale (CNN), potrebbero essere determinate istanze di filtro o canali. Per reti completamente connesse, potrebbero essere i neuroni stessi mantenendo la piena connettività o persino l'eliminazione di interi layer del modello che sono considerati insignificanti. Questo tipo di potatura spesso porta a reti sparse regolari e strutturate che sono compatibili con l'hardware.
Sono iniziate a emergere le "best practice" su come pensare alla potatura strutturata. Ci sono tre componenti principali:

9.2.1.2.1 1. Strutture Candidate per il Pruning

Data la varietà di approcci, diverse strutture all'interno di una rete neurale vengono potate in base a criteri specifici. Le strutture primarie per la potatura includono neuroni, canali e talvolta interi layer, ognuno con le sue implicazioni e metodologie uniche. L'obiettivo di ogni approccio è garantire che il modello ridotto mantenga il più possibile la capacità predittiva del modello originale, migliorando al contempo l'efficienza computazionale e riducendo le dimensioni.

Quando i **neuroni** vengono potati, rimuoviamo interi neuroni insieme ai loro pesi e bias associati, riducendo così la larghezza del layer. Questo tipo di potatura viene spesso utilizzato in layer completamente connessi.

La potatura del **canale**, che viene applicata prevalentemente nelle reti neurali convoluzionali (CNN), comporta l'eliminazione di interi canali o filtri, il che a sua volta riduce la profondità delle mappe delle feature e influisce sulla capacità della rete di estrarre determinate feature dai dati di input. Ciò è particolarmente cruciale nelle attività di elaborazione delle immagini in cui l'efficienza computazionale è fondamentale.

Infine, la potatura dei **layer** adotta un approccio più aggressivo rimuovendo interi layer della rete. Ciò riduce significativamente la profondità della rete e quindi la sua capacità di plasmare pattern e gerarchie complesse nei dati. Questo approccio richiede un attento equilibrio per garantire che la capacità predittiva del modello non venga indebitamente compromessa.
Figura 9.2 mostra la differenza tra la potatura di canale/filtro e quella del layer. Quando potiamo un canale, dobbiamo riconfigurare l'architettura del modello per adattarla ai cambiamenti strutturali. Una modifica consiste nel cambiare il numero di canali di input nel layer successivo (qui, il terzo e il layer più profondo): modificando le profondità dei filtri applicati al layer con il canale potato. D'altra parte, la potatura di un intero layer (rimuovendo tutti i canali nel layer) richiede modifiche più drastiche. Quella principale riguarda la modifica delle connessioni tra i layer rimanenti per sostituire o bypassare il layer potato. Nel nostro

caso, riconfiguriamo per connettere il primo e l'ultimo layer. In tutti i casi di potatura, dobbiamo mettere a punto la nuova struttura per regolare i pesi.

Figura 9.2: Potatura del canale e quella del layer.

9.2.1.2.2 2. Stabilire un criterio per il Pruning

Stabilire criteri ben definiti per determinare quali strutture specifiche potare da un modello di rete neurale è una componente cruciale del processo di "pruning" del modello. L'obiettivo principale qui è identificare e rimuovere i componenti che contribuiscono meno alle capacità predittive del modello, mantenendo al contempo le strutture integrali per preservare l'accuratezza.

Una strategia ampiamente adottata ed efficace per potare sistematicamente le strutture si basa sul calcolo di punteggi di importanza per singoli componenti come neuroni, filtri, canali o layer. Questi punteggi servono come metriche quantitative per valutare la significatività di ciascuna struttura e il suo effetto sull'output del modello.

Esistono diverse tecniche per assegnare questi punteggi sull'importanza:

- **Pruning Basato sulla Magnitudo del peso**: Questo approccio assegna punteggi di importanza a una struttura valutando la magnitudo aggregata dei pesi associati. Le strutture con magnitudo del peso complessivo inferiore sono considerate meno critiche per le prestazioni della rete.
- **Pruning Basato sul Bradiente**: Questa tecnica utilizza i gradienti della funzione di los [perdita] rispetto ai pesi associati a una struttura. Le strutture con magnitudo del gradiente cumulativo basso, che indica un impatto minimo sulla perdita quando alterato, sono le candidate principali per la potatura.

- **Pruning Basato sull'Attivazione**: Questo metodo tiene traccia della frequenza con cui un neurone o un filtro viene attivato memorizzando queste informazioni in un parametro chiamato contatore delle attivazioni. Ogni volta che la struttura viene attivata, il contatore viene incrementato. Un conteggio di attivazione basso suggerisce che la struttura è meno rilevante.
- **Pruning Basato sull'Espansione di Taylor**: Questo approccio approssima la modifica nella funzione di perdita derivante dalla rimozione di un dato peso. Valutando la perturbazione della perdita cumulativa derivante dalla rimozione di tutti i pesi associati a una struttura, è possibile identificare le strutture con un impatto trascurabile sulla perdita, rendendole candidate idonee per la potatura.

L'idea è di misurare, direttamente o indirettamente, il contributo di ogni componente all'output del modello. Le strutture con un'influenza minima in base ai criteri definiti vengono potate per prime. Ciò consente una potatura selettiva e ottimizzata che comprime al massimo i modelli preservando al contempo la capacità predittiva. In generale, è importante valutare l'impatto della rimozione di particolari strutture sull'output del modello, con lavori recenti come (Rachwan et al. 2022) e (Lubana e Dick 2020) che studiano combinazioni di tecniche come la potatura basata sulla magnitudine e la potatura basata sul gradiente.

9.2.1.2.3 3. Selezione di una Strategia di Potatura

Ora che abbiamo capito alcune tecniche per determinare l'importanza delle strutture all'interno di una rete neurale, il passo successivo è decidere come applicare queste intuizioni. Ciò comporta la selezione di una strategia di potatura appropriata, che stabilisce come e quando le strutture identificate vengono rimosse e come il modello viene messo a punto per mantenere le sue prestazioni. Esistono due principali strategie di potatura strutturata: quella iterativa e la one-shot.

La **potatura iterativa** rimuove gradualmente le strutture attraverso più cicli di potatura seguiti da messa a punto. In ogni ciclo, un piccolo set di strutture viene potato in base a criteri di importanza. Il modello viene poi messo a punto, consentendogli di adattarsi senza problemi ai cambiamenti strutturali prima della successiva iterazione di potatura. Questo approccio graduale e ciclico impedisce bruschi cali di accuratezza. Consente al modello di adattarsi lentamente man mano che le strutture vengono ridotte attraverso le iterazioni. Consideriamo una situazione in cui desideriamo potare i 6 canali meno efficaci (in base ad alcuni criteri specifici) da una rete neurale convoluzionale. In Figura 9.3, mostriamo un processo di potatura semplificato eseguito su 3 iterazioni. In ogni iterazione, eliminiamo solo 2 canali. La rimozione dei canali comporta un degrado della precisione. Nella prima iterazione, la precisione scende da 0.995 a 0.971. Tuttavia, dopo aver perfezionato il modello sulla nuova struttura, siamo in grado di recuperare dalla perdita di prestazioni, portando la precisione a 0.992. Poiché i cambiamenti strutturali sono minori e graduali, la rete può adattarsi più facilmente a essi. Eseguendo lo stesso processo altre 2 volte, finiamo con una precisione finale di 0.991 (una perdita di solo lo 0.4% rispetto all'originale) e una riduzione del 27% nel numero di canali. Pertanto, la potatura iterativa ci consente di mantenere le prestazioni beneficiando di una maggiore efficienza computazionale dovuta alla riduzione delle dimensioni del modello.

Figura 9.3: Potatura iterativa.

La **potatura one-shot** adotta un approccio più aggressivo, potando una grande porzione di strutture simultaneamente in un'unica operazione in base a criteri di importanza predefiniti. Segue un'ampia messa a punto per recuperare l'accuratezza del modello. Sebbene più rapida, questa strategia aggressiva può degradare l'accuratezza se il modello non riesce a recuperare durante la messa a punto.

La scelta tra queste strategie comporta la valutazione di fattori quali dimensioni del modello, quanto è sparso il target, calcolo disponibile e perdite di accuratezza accettabili. La potatura one-shot può comprimere rapidamente i modelli, ma quella iterativa può consentire una migliore conservazione dell'accuratezza per un livello target di potatura. In pratica, la strategia è personalizzata in base ai vincoli del caso d'uso. L'obiettivo generale è quello di generare una strategia ottimale che rimuova la ridondanza, ottenga guadagni di efficienza tramite la potatura e metta a punto il modello per stabilizzare l'accuratezza a un livello accettabile per l'implementazione.

Ora si consideri la stessa rete che avevamo nell'esempio di potatura iterativa. Mentre nel processo iterativo abbiamo potato 2 canali alla volta, nel processo one-shot poteremo i 6 canali contemporaneamente, come mostrato in Figura 9.4. La rimozione simultanea del 27% del canale della rete altera significativamente la struttura, causando un calo della precisione da 0.995 a 0.914. Date le modifiche principali, la rete non è in grado di adattarsi correttamente durante la messa a punto e la precisione è salita a 0.943, un degrado del 5% rispetto alla precisione della rete non potata. Mentre le strutture finali nei processi di potatura iterativa e di potatura one-shot sono identiche, la prima è in grado di mantenere prestazioni elevate mentre la seconda subisce degradi significativi.

Figura 9.4: Potatura one-shot.

9.2.1.3 Vantaggi della Potatura Strutturata

La potatura strutturata offre una miriade di vantaggi che soddisfano vari aspetti dell'implementazione e dell'utilizzo del modello, specialmente in ambienti in cui le risorse computazionali sono limitate.

- **Efficienza Computazionale:** Eliminando intere strutture, come neuroni o canali, si riduce significativamente il carico computazionale durante le fasi di training e inferenza, consentendo così previsioni più rapide del modello e convergenza del training. Inoltre, la rimozione delle strutture riduce intrinsecamente il "footprint" [impronta] di memoria del modello, assicurando che richieda meno spazio di archiviazione e memoria durante il funzionamento, il che è particolarmente vantaggioso in ambienti con limiti di memoria come i sistemi TinyML.
- **Efficienza Hardware:** La potatura strutturata spesso si traduce in modelli più adatti all'implementazione su hardware specializzato, come i Field-Programmable Gate Arrays (FPGA) o Application-Specific Integrated Circuits (ASIC), a causa della regolarità e la semplicità dell'architettura potata. Con requisiti di elaborazione ridotti, si traduce in un consumo energetico inferiore, fondamentale per i dispositivi alimentati a batteria e i metodi di elaborazione sostenibili.
- **Manutenzione e Distribuzione:** Il modello ridotto, sebbene più piccolo, mantiene la sua forma architettonica originale, che può semplificare la pipeline di distribuzione e garantire la compatibilità con i sistemi e i framework esistenti. Inoltre, con meno parametri e strutture più semplici, il modello potato diventa più facile da gestire e monitorare negli ambienti di produzione, riducendo potenzialmente le spese generali associate alla manutenzione e agli aggiornamenti del modello. Più avanti, quando approfondiremo MLOps, questa necessità diventerà evidente.

9.2.1.4 Potatura non Strutturata

Il "pruning" non-strutturato è, come suggerisce il nome, la potatura del modello senza riguardo alla sotto-struttura specifica del modello. Come accennato in precedenza, offre una maggiore aggressività nella potatura e può raggiungere maggiori diradazione del modello mantenendo la precisione, dati meno vincoli su ciò che può e non può essere potato. In genere, la potatura non-strutturata post-training consiste in un criterio di importanza per i singoli parametri/pesi del modello, potatura/rimozione dei pesi che scendono al di sotto dei criteri e una successiva messa a punto facoltativa per provare a recuperare la precisione persa durante la rimozione dei pesi.

La potatura non-strutturata presenta alcuni vantaggi rispetto a quella strutturata: la rimozione di singoli pesi anziché di intere sotto-strutture del modello spesso porta in pratica a minori diminuzioni della precisione del modello. Inoltre, in genere determinare il criterio di importanza per un singolo peso è molto più semplice che per un'intera sotto-struttura di parametri nella potatura strutturata, rendendo la prima preferibile nei casi in cui tale overhead è difficile o poco chiaro da calcolare. Analogamente, il processo effettivo di

potatura strutturata è generalmente meno flessibile, poiché la rimozione di singoli pesi è generalmente più semplice della rimozione di intere sotto-strutture e della garanzia che il modello funzioni ancora.
La potatura non strutturata, pur offrendo il potenziale per una significativa riduzione delle dimensioni del modello e una migliore implementabilità, porta con sé problemi legati alla gestione di rappresentazioni sparse e alla garanzia dell'efficienza computazionale. È particolarmente utile in scenari in cui è fondamentale ottenere la massima compressione possibile del modello e in cui l'ambiente di distribuzione può gestire in modo efficiente i calcoli sparsi.
Tabella 9.1 fornisce un confronto conciso tra potatura strutturata e la non-strutturata. In questa tabella, gli aspetti relativi alla natura e all'architettura del modello potato (Definizione, Regolarità del modello e Livello di compressione) sono raggruppati insieme, seguiti dagli aspetti relativi alle considerazioni computazionali (Efficienza computazionale e Compatibilità hardware) e terminando con gli aspetti relativi all'implementazione e all'adattamento del modello potato (Complessità di implementazione e Complessità di messa a punto).
Entrambe le strategie di potatura offrono vantaggi e problemi unici, come mostrato in Tabella 9.1, e la selezione tra di esse dovrebbe essere influenzata da requisiti specifici del progetto e della distribuzione.

Tabella 9.1: Confronto tra potatura strutturata e non-strutturata.

Aspetto	Potatura strutturata	Potatura non strutturata
Definizione	Potatura di intere strutture (ad esempio, neuroni, canali, layer) all'interno della rete	Potatura di singoli pesi o neuroni, con conseguenti matrici sparse o strutture di rete non regolari
Regolarità del Modello	Mantiene un'architettura di rete regolare e strutturata	Si traduce in architetture di rete irregolari e sparse
Livello di Compressione	Può offrire una compressione del modello limitata rispetto alla potatura non-strutturata	Può ottenere una compressione del modello più elevata grazie alla potatura a grana fine
Efficienza Computazionale	In genere più efficiente computazionalmente grazie al mantenimento di strutture regolari	Può essere inefficiente dal punto di vista computazionale a causa di matrici di peso sparse, a meno che non venga utilizzato hardware/software specializzato
Compatibilità Hardware	In genere più compatibile con vari hardware grazie alle strutture regolari	Potrebbe richiedere hardware che gestisca in modo efficiente i calcoli sparsi per ottenere vantaggi
Complessità di Implementazione	Spesso più semplice da implementare e gestire grazie al mantenimento della struttura della rete	Può essere complesso da gestire e calcolare a causa delle rappresentazioni sparse
Complessità di Messa a Punto Fine	Potrebbe richiedere strategie di messa a punto fine meno complesse dopo la potatura	Potrebbe richiedere strategie di riaddestramento o messa a punto fine più complesse dopo la potatura

In Figura 9.5 abbiamo esempi che illustrano le differenze tra potatura non-strutturata e strutturata. Osservare che la potatura non-strutturata può portare a modelli che non rispettano più le garanzie strutturali di alto livello delle loro controparti originali non potate: la rete di sinistra non è più una rete completamente connessa dopo la potatura. La potatura strutturata, d'altro canto, mantiene quelle invarianti: al centro, la rete completamente connessa viene potata in modo che resti ancora completamente connessa; allo stesso modo, la CNN mantiene la sua struttura convoluzionale, sebbene con meno filtri.

Figura 9.5: Potatura non-strutturata e strutturata. Fonte: Qi et al. (2021).

9.2.1.5 Ipotesi del Biglietto della Lotteria

La potatura si è evoluta da una tecnica puramente post-addestramento che comportava un costo per una certa accuratezza, a un potente approccio di meta-apprendimento applicato durante l'addestramento per ridurre la complessità del modello. Questo progresso a sua volta migliora l'efficienza di calcolo, memoria e latenza sia nell'addestramento che nell'inferenza.

Una scoperta rivoluzionaria che ha catalizzato questa evoluzione è stata l'ipotesi del biglietto della lotteria di Frankle e Carbin (2019). Il loro lavoro afferma che all'interno di reti neurali dense esistono sotto-reti sparse, denominate "biglietti vincenti", che possono eguagliare o addirittura superare le prestazioni del modello originale quando addestrate in isolamento. In particolare, questi biglietti vincenti, quando inizializzati utilizzando gli stessi pesi della rete originale, possono raggiungere una convergenza e un'accuratezza di addestramento altrettanto elevate su un dato compito. Vale la pena sottolineare che hanno scoperto empiricamente l'ipotesi del biglietto della lotteria, che è stata successivamente formalizzata. L'intuizione alla base di questa ipotesi è che, durante il processo di addestramento di una rete neurale, molti neuroni e connessioni diventano ridondanti o non importanti, in particolare con l'inclusione di tecniche di addestramento che incoraggiano la ridondanza come il "dropout" [abbandono]. L'identificazione, la potatura e l'inizializzazione di questi "biglietti vincenti" consentono un addestramento più rapido e modelli più efficienti, poiché contengono le informazioni essenziali per la decisione del modello per l'attività. Inoltre, come generalmente noto con la teoria del "bias-variance tradeoff" [compromesso tra bias e varianza], questi biglietti soffrono meno di sovra-parametrizzazione e quindi si generalizzano meglio piuttosto che sovra-adattarsi all'attività.

In Figura 9.6 abbiamo un esempio che mostra esperimenti di potatura e addestramento su una LeNet completamente connessa su una varietà di rapporti di potatura. Nel grafico a sinistra, si nota come una potatura pesante riveli una sotto-rete più efficiente (in verde) che è il 21,1% delle dimensioni della rete originale (in blu). La sotto-rete raggiunge una maggiore accuratezza e in modo più rapido rispetto alla versione non potata (la linea verde è sopra la linea blu). Tuttavia, la potatura ha un limite (punto ottimale) e un'ulteriore potatura produrrà degradi delle prestazioni e alla fine scenderà al di sotto delle prestazioni della versione non potata (nota come le sotto-reti rossa, viola e marrone diminuiscono gradualmente nelle prestazioni di accuratezza) a causa della significativa perdita nel numero di parametri.

Figura 9.6: Esperimenti sull'ipotesi del biglietto della lotteria.

Per scoprire questi biglietti vincenti della lotteria all'interno di una rete neurale, viene seguito un processo sistematico. Questo processo, illustrato in Figura 9.7 (a sinistra), prevede l'addestramento iterativo, la potatura e la reinizializzazione della rete. I passaggi seguenti delineano questo approccio:

1. Inizializzare i pesi della rete a valori casuali.
2. Addestrare la rete finché non converge alle prestazioni desiderate.
3. Eliminare una percentuale di rami con i valori di peso più bassi.
4. Reinizializzare la rete con gli stessi valori casuali del passaggio 1.
5. Ripetere i passaggi 2-4 più volte o finché la precisione non peggiora in modo significativo.

Alla fine, ci si ritrova con una rete potata (Figura 9.7 lato destro), che è una sotto-rete di quella di partenza. La sotto-rete dovrebbe avere una struttura significativamente più piccola, pur mantenendo un livello di precisione comparabile.

Figura 9.7: Trovare la sottorete del biglietto vincente.

9.2.1.6 Problemi e Limitazioni

Non c'è niente di gratuito con le ottimizzazioni di potatura, con alcune scelte che comportano sia miglioramenti che costi da considerare. Di seguito, discutiamo alcuni compromessi che gli esperti devono considerare.

- **Gestione di Matrici di Peso Sparse:** Una matrice di peso sparsa è una matrice in cui molti degli elementi sono pari a zero. La potatura non strutturata spesso produce

matrici di peso sparse, in cui molti pesi vengono potati a zero. Sebbene ciò riduca le dimensioni del modello, introduce anche diversi problemi. L'inefficienza computazionale può sorgere perché l'hardware standard è ottimizzato per operazioni di matrice densa. Senza ottimizzazioni che sfruttano la sparsità, i risparmi computazionali derivanti dalla potatura possono essere persi. Sebbene le matrici sparse possano essere archiviate senza formati specializzati, sfruttare efficacemente la loro sparsità richiede una gestione attenta per evitare di sprecare risorse. Algoritmicamente, la navigazione in strutture sparse richiede di saltare in modo efficiente le voci zero, il che aggiunge complessità al calcolo e agli aggiornamenti del modello.

- **Qualità vs. Riduzione delle Dimensioni:** Una sfida fondamentale sia nella potatura strutturata che in quella non-strutturata è bilanciare la riduzione delle dimensioni con il mantenimento o il miglioramento delle prestazioni predittive. È essenziale stabilire criteri di potatura robusti, sia per rimuovere intere strutture (potatura strutturata) sia singoli pesi (potatura non strutturata). Questi criteri di potatura scelti devono identificare accuratamente gli elementi la cui rimozione ha un impatto minimo sulle prestazioni. Spesso è necessaria un'attenta sperimentazione per garantire che il modello potato rimanga efficiente mantenendo al contempo le sue prestazioni predittive.
- **Fine-Tuning e Riaddestramento:** La messa a punto post-potatura è fondamentale sia nella potatura strutturata che in quella non-strutturata per recuperare le prestazioni perse e stabilizzare il modello. La sfida comprende la determinazione dell'estensione, della durata e della natura del processo di messa a punto, che può essere influenzato dal metodo di potatura e dal grado di potatura applicato.
- **Compatibilità ed Efficienza Hardware:** Particolarmente pertinenti alla potatura non-strutturata, la compatibilità e l'efficienza hardware diventano critiche. La potatura non strutturata spesso si traduce in matrici di peso sparse, che potrebbero non essere gestite in modo efficiente da un certo hardware, annullando potenzialmente i vantaggi computazionali della potatura (vedere Figura 9.8). Garantire che i modelli potati, in particolare quelli risultanti dall'eliminazione non-strutturata, siano scalabili, compatibili ed efficienti sull'hardware target è una considerazione importante.
- **Considerazioni Legali ed Etiche:** Ultimo ma non meno importante, il rispetto delle linee guida legali ed etiche è importante, soprattutto in ambiti con conseguenze significative. I metodi di potatura devono essere sottoposti a rigorosi processi di validazione, test e potenzialmente certificazione per garantire la conformità alle normative e agli standard pertinenti, sebbene al momento non esistano standard formali e "best practice" che siano esaminati e convalidati da entità terze. Ciò è particolarmente cruciale in applicazioni ad alto rischio come l'intelligenza artificiale medica e la guida autonoma, dove i cali di qualità dovuti a ottimizzazioni simili alla potatura possono essere pericolosi per la vita. Inoltre, le considerazioni etiche si estendono oltre la sicurezza fino all'equità e all'uguaglianza; un recente lavoro di (Tran et al. 2022) ha rivelato che la potatura può avere un impatto sproporzionato sulle persone di colore, sottolineando la necessità di una valutazione etica completa nel processo di potatura.

Figura 9.8: Matrice dei pesi sparsi.

Esercizio 9.1: Pruning

Si immagini che la rete neurale sia un cespuglio gigante e troppo cresciuto. La potatura è come tagliare strategicamente i rami per renderla più forte ed efficiente! Nel Colab, si imparerà come fare questa potatura in TensorFlow. La comprensione di questi concetti fornirà le basi per vedere come la potatura rende i modelli abbastanza piccoli da poter essere eseguiti sul telefono!

Open in Colab

9.2.2 Compressione del Modello

Le tecniche di compressione del modello sono fondamentali per distribuire modelli di deep learning su dispositivi con risorse limitate. Queste tecniche mirano a creare modelli più piccoli ed efficienti che preservino le prestazioni predittive dei modelli originali.

9.2.2.1 Distillazione della Conoscenza

Una tecnica popolare è la **knowledge distillation (KD)** distillazione della conoscenza, che trasferisce la conoscenza da un modello "insegnante" ampio e complesso a un modello "studente" più piccolo. L'idea chiave è addestrare il modello studente a imitare gli output dell'insegnante. Il concetto di KD è stato reso popolare per la prima volta da Hinton (2005).

9.2.2.1.1 Panoramica e Vantaggi

La distillazione della conoscenza implica il trasferimento della conoscenza da un modello insegnante ampio e complesso a un modello studente più piccolo. L'idea di base è quella di utilizzare gli output dell'insegnante, noti come **soft targets**, per guidare il training del modello studente. A differenza dei tradizionali "hard targets" (le vere etichette), quelli soft sono le distribuzioni di probabilità sulle classi che il modello insegnante prevede. Queste distribuzioni forniscono informazioni più complete sulle relazioni tra le classi, il che può aiutare il modello studente ad apprendere in modo più efficace.

Abbiamo imparato che la funzione softmax converte gli output grezzi di un modello in una distribuzione di probabilità sulle classi. Una tecnica chiave in KD è la **scalatura della**

temperatura, che viene applicata alla funzione softmax degli output del modello insegnante. Introducendo un parametro di temperatura, la distribuzione può essere regolata: una temperatura più alta produce probabilità più soft, il che significa che le differenze tra le probabilità di classe diventano meno estreme. Questo effetto di ammorbidimento determina una distribuzione più uniforme, in cui la fiducia del modello nella classe più probabile è ridotta e altre classi hanno probabilità più elevate, diverse da zero. Ciò è prezioso per il modello studente perché gli consente di apprendere non solo dalla classe più probabile, ma anche dalle probabilità relative di tutte le classi, catturando pattern sottili che potrebbero essere persi se addestrati solo su obiettivi difficili. Pertanto, la scalabilità della temperatura facilita il trasferimento di conoscenze più sfumate dal modello insegnante a quello studente.
La funzione di perdita nella distillazione della conoscenza in genere combina due componenti: una perdita di distillazione e una perdita di classificazione. La perdita di distillazione, spesso calcolata utilizzando la divergenza di Kullback-Leibler (KL), misura la differenza tra gli soft target prodotti dal modello insegnante e gli output del modello studente, incoraggiando lo studente a imitare le previsioni dell'insegnante. Nel frattempo, la perdita di classificazione assicura che il modello studente preveda correttamente le etichette vere in base ai dati originali. Insieme, queste due componenti aiutano lo studente modello a conservare le conoscenze dell'insegnante, rispettando al contempo le etichette di verità di base.
Questi componenti, quando configurati e armonizzati abilmente, consentono al modello studente di assimilare la conoscenza del modello insegnante, creando un percorso verso modelli più piccoli, efficienti e robusti, che mantengono la capacità predittiva delle loro controparti più grandi. Figura 9.9 visualizza la procedura di training della "knowledge distillation". Notare come i logit o le soft label del modello insegnante vengono utilizzati per fornire una perdita di distillazione da cui il modello studente può imparare.

Figura 9.9: Processo di training della distillazione della conoscenza. Fonte: IntelLabs (2023).

9.2.2.1.2 Sfide

Tuttavia, KD presenta una serie unica di sfide e considerazioni che ricercatori e professionisti devono affrontare attentamente. Una delle sfide è nella messa a punto meticolosa degli iperparametri, come il parametro "temperatura" nella funzione softmax e la ponderazione tra la distillazione e la perdita di classificazione nella funzione obiettivo. Raggiungere un equilibrio che sfrutti efficacemente gli output ammorbiditi del modello insegnante mantenendo al contempo la fedeltà alle etichette dei dati reali non è banale e può avere un impatto significativo sulle prestazioni e sulle capacità di generalizzazione del modello studente.

Inoltre, l'architettura del modello studente stesso pone un problema considerevole. Progettare un modello compatto per soddisfare i vincoli di calcolo e memoria, pur essendo in grado di assimilare le conoscenze essenziali dal modello insegnante, richiede una comprensione sfumata della capacità del modello e dei compromessi intrinseci coinvolti nella compressione. Il modello studente deve essere attentamente progettato per navigare nella dicotomia di dimensioni e prestazioni, assicurando che la conoscenza distillata venga catturata e utilizzata in modo significativo. Inoltre, la scelta del modello dell'insegnante, che influenza intrinsecamente la qualità e la natura della conoscenza da trasferire, è importante e introduce un ulteriore livello di complessità nel processo KD.
Queste sfide sottolineano la necessità di un approccio completo e sfumato all'implementazione di KD, assicurando che i modelli degli studenti risultanti siano sia efficienti che efficaci nei loro contesti operativi.

9.2.2.2 Fattorizzazione di Matrici di Basso Rango

Simile nel tema dell'approssimazione, la Low-Rank Matrix Factorization (LRMF) fattorizzazione di matrici di basso rango è una tecnica matematica utilizzata in algebra lineare e analisi dei dati per approssimare una matrice data scomponendola in due o più matrici di dimensione inferiore. L'idea fondamentale è di esprimere una matrice di grandi dimensioni come prodotto di matrici di rango inferiore, il che può aiutare a ridurre la complessità dei dati preservandone la struttura essenziale. Matematicamente, data una matrice $A \in \mathbb{R}^{m \times n}$, LRMF cercare le matrici $U \in \mathbb{R}^{m \times k}$ e $V \in \mathbb{R}^{k \times n}$ tali che $A \approx UV$, dove k è il rango ed è in genere molto più piccolo di m e n.

9.2.2.2.1 Background e Benefici

Uno dei primo lavori nel campo della fattorizzazione di matrici, in particolare nel contesto dei sistemi di raccomandazione, è il documento di Koren, Bell, e Volinsky (2009). Gli autori esaminano vari modelli di fattorizzazione, fornendo approfondimenti sulla loro efficacia nel catturare i pattern sottostanti nei dati e nel migliorare l'accuratezza predittiva nel filtraggio collaborativo. LRMF è stato ampiamente applicato nei sistemi di raccomandazione (come Netflix, Facebook, ecc.), dove la matrice di interazione utente-elemento è fattorizzata per catturare fattori latenti corrispondenti alle preferenze dell'utente e agli attributi dell'elemento.
Il vantaggio principale della "fattorizzazione di matrici di basso rango" risiede nella sua capacità di ridurre la dimensionalità dei dati come mostrato in Figura 9.10, dove ci sono meno parametri da memorizzare, rendendola più efficiente dal punto di vista computazionale e riducendo i requisiti di archiviazione a costo di un po' di elaborazione aggiuntiva. Ciò può portare a calcoli più rapidi e rappresentazioni di dati più compatte, il che è particolarmente prezioso quando si ha a che fare con grandi set di dati. Inoltre, può aiutare nella riduzione del rumore e può rivelare pattern e relazioni sottostanti nei dati.
Figura 9.10 illustra la diminuzione della parametrizzazione abilitata dalla fattorizzazione di matrici di basso rango. Osservare come la matrice M può essere approssimata dal prodotto delle matrici L_k e R_k^T. Per intuizione, la maggior parte dei layer completamente connessi nelle reti sono archiviati come matrice di proiezione M, che richiede il caricamento di $m \times n$ parametri durante il calcolo. Tuttavia, scomponendola e approssimandola come prodotto di due matrici di rango inferiore, abbiamo bisogno di archiviare solo $m \times k + k \times n$ parametri in termini di archiviazione, sostenendo al contempo un costo di calcolo aggiuntivo per la moltiplicazione delle matrici. Finché $k < n/2$, questa fattorizzazione ha meno parametri totali da archiviare, aggiungendo un calcolo di runtime $O(mkn)$ (Gu 2023).

Figura 9.10: Fattorizzazione di matrici di basso rango. Fonte: The Clever Machine.

9.2.2.2.2 Sfide

Ma professionisti e ricercatori incontrano una serie di problemi e considerazioni che richiedono una particolare attenzione e approcci strategici. Come con qualsiasi tecnica di compressione lossy [con perdita], potremmo perdere informazioni durante questo processo di approssimazione: scegliere il rango corretto che bilanci le informazioni perse e i costi computazionali è altrettanto complicato e aggiunge un ulteriore iperparametro da regolare.
La fattorizzazione di matrici di basso rango è uno strumento prezioso per la riduzione della dimensionalità e per adattare il calcolo ai dispositivi edge ma, come altre tecniche, deve essere attentamente regolata in base al modello e all'attività da svolgere. Una sfida fondamentale risiede nella gestione della complessità computazionale inerente a LRMF, soprattutto quando si hanno a che fare con dati ad alta dimensionalità e su larga scala. L'onere computazionale, in particolare nel contesto di applicazioni in tempo reale e set di dati massicci, rimane un ostacolo significativo per un utilizzo efficace di LRMF.
Inoltre, l'enigma della scelta del rango ottimale k per la fattorizzazione introduce un ulteriore livello di complessità. La selezione di k implica intrinsecamente un compromesso tra accuratezza dell'approssimazione e semplicità del modello, e l'identificazione di un rango che bilanci abilmente questi obiettivi contrastanti spesso richiede una combinazione di competenza di dominio, convalida empirica e, a volte, approcci euristici. La sfida è ulteriormente amplificata quando i dati comprendono rumore o quando la struttura intrinseca di basso rango non è pronunciata, rendendo la determinazione di un k adatto ancora più sfuggente.
La gestione di dati mancanti o sparsi, un evento comune in applicazioni come i sistemi di raccomandazione, pone un'altra sfida sostanziale. Le tecniche tradizionali di fattorizzazione delle matrici, come la Singular Value Decomposition (SVD), non sono direttamente applicabili alle matrici con voci mancanti, rendendo necessario lo sviluppo e l'applicazione di algoritmi specializzati in grado di fattorizzare matrici incomplete mitigando al contempo i rischi di overfitting alle voci osservate. Ciò spesso comporta l'incorporazione di termini di regolarizzazione o la limitazione della fattorizzazione in modi specifici, il che a sua volta introduce ulteriori iperparametri che devono essere selezionati giudiziosamente.
Inoltre, in scenari in cui i dati evolvono o crescono nel tempo, sviluppare modelli LRMF in grado di adattarsi a nuovi dati senza richiedere una completa rifattorizzazione è un'impresa critica ma impegnativa. Gli algoritmi di fattorizzazione di matrici incrementali e online cercano di risolvere questo problema consentendo l'aggiornamento delle matrici fattorizzate all'arrivo di nuovi dati, ma garantire stabilità, accuratezza ed efficienza computazionale in queste impostazioni dinamiche rimane un compito intricato. Ciò è particolarmente

impegnativo nello spazio di TinyML, in cui la ridistribuzione dei rami per i modelli aggiornati può essere piuttosto impegnativa.

9.2.2.3 Decomposizione dei Tensori

Abbiamo visto in Sezione 6.4.1 che i tensori sono strutture flessibili, comunemente utilizzate dai framework ML, che possono rappresentare dati in dimensioni superiori. Similmente alla fattorizzazione di matrici di basso rango, i modelli più complessi possono memorizzare pesi in dimensioni superiori, come i tensori. La decomposizione tensoriale è l'analogo a più dimensioni della fattorizzazione della matrice, in cui un tensore modello viene scomposto in componenti di rango inferiore (Figura 9.11). Questi componenti di rango inferiore sono più facili da calcolare e memorizzare, ma possono soffrire degli stessi problemi menzionati sopra, come la perdita di informazioni e la necessità di una messa a punto sfumata degli iperparametri. Matematicamente, dato un tensore $\mathcal{A}$, la decomposizione tensoriale cerca di rappresentare $\mathcal{A}$ come una combinazione di tensori più semplici, facilitando una rappresentazione compressa che approssima i dati originali riducendo al minimo la perdita di informazioni.

Figura 9.11: Decomposizione dei Tensori. Fonte: Xinyu (s.d.).

Il lavoro di Tamara G. Kolda e Brett W. Bader, "Tensor Decompositions and Applications" (2009), si distingue come un articolo fondamentale nel campo delle decomposizioni tensoriali. Gli autori forniscono una panoramica completa di vari metodi di decomposizione tensoriale, esplorandone i fondamenti matematici, gli algoritmi e un'ampia gamma di applicazioni, che vanno dall'elaborazione del segnale al data mining. Naturalmente, il motivo per cui ne stiamo discutendo è perché ha un enorme potenziale per i miglioramenti delle prestazioni del sistema, in particolare nello spazio di TinyML, dove la produttività e i risparmi di memoria sono fondamentali per la fattibilità delle distribuzioni.

Esercizio 9.2: Compressione di Modelli Scalabili con TensorFlow

Questo Colab si addentra in una tecnica per comprimere i modelli mantenendo un'elevata accuratezza. L'idea chiave è quella di addestrare un modello con un termine di penalità extra che incoraggia il modello a essere più comprimibile. Quindi, il modello viene codificato utilizzando uno schema di codifica speciale che si allinea con questa penalità. Questo approccio consente di ottenere modelli compressi che funzionano altrettanto bene dei modelli originali ed è utile per distribuire modelli su dispositivi con risorse limitate come telefoni cellulari e dispositivi edge.

Open in Colab

9.2.3 Modelli Progettati per l'Edge

Ora raggiungiamo l'altro estremo del gradiente hardware-software, dove prendiamo decisioni specifiche sull'architettura del modello direttamente in base alla conoscenza dei dispositivi edge su cui desideriamo implementare.

Come spiegato nelle sezioni precedenti, i dispositivi edge sono vincolati specificamente da limitazioni di memoria e calcoli parallelizzabili: in quanto tali, se ci sono requisiti critici di velocità di inferenza, i calcoli devono essere sufficientemente flessibili da soddisfare i vincoli hardware, qualcosa che può essere progettato a livello di architettura del modello. Inoltre, cercare di stipare grandi modelli SOTA ML su dispositivi edge anche dopo potatura e compressione è generalmente irrealizzabile puramente a causa delle dimensioni: la complessità del modello stesso deve essere scelta con più sfumature per adattarsi più fattibilmente al dispositivo. Gli sviluppatori di Edge ML hanno affrontato questa sfida architettonica sia attraverso la progettazione di architetture di modelli edge ML su misura sia attraverso la Neural Architecture Search (NAS) [ricerca di architettura neurale] avente il dispositivo come target, che può generare in modo più sistematico architetture fattibili di modelli su dispositivo.

9.2.3.1 Tecniche di Progettazione del Modello

Un design di architettura edge friendly, comunemente utilizzato nel deep learning per l'elaborazione delle immagini, è quello delle convoluzioni separabili in profondità. Consiste in due fasi distinte: la prima è la convoluzione in profondità, in cui ogni canale di input viene convoluto in modo indipendente con il proprio set di filtri apprendibili, come mostrato in Figura 9.12. Questa fase riduce la complessità computazionale in modo significativo rispetto alle convoluzioni standard, poiché riduce drasticamente il numero di parametri e calcoli coinvolti. La seconda fase è la convoluzione puntuale, che combina l'output dei canali di convoluzione in profondità tramite una convoluzione 1x1, creando interazioni tra canali. Questo approccio offre diversi vantaggi. I vantaggi includono dimensioni ridotte del modello, tempi di inferenza più rapidi e spesso una migliore generalizzazione grazie al minor numero di parametri, rendendolo adatto ad applicazioni mobili ed embedded. Tuttavia, le convoluzioni separabili in profondità potrebbero non catturare interazioni spaziali complesse in modo efficace come le convoluzioni standard e potrebbero richiedere più profondità (livelli) per raggiungere lo stesso livello di potenza rappresentativa, portando potenzialmente a tempi di addestramento più lunghi. Tuttavia, la loro efficienza in termini di parametri e calcolo le rende una scelta popolare nelle moderne architetture di reti neurali convoluzionali.

Figura 9.12: Convoluzioni separabili in profondità. Fonte: Hegde (2023).

9.2.3.2 Architetture di Modello di Esempio

In quest'ottica, diverse architetture recenti sono state, fin dall'inizio, progettate specificamente per massimizzare la precisione in un'implementazione edge, in particolare SqueezeNet, MobileNet ed EfficientNet.

- SqueezeNet di Iandola et al. (2016), ad esempio, utilizza un'architettura compatta con convoluzioni 1x1 e moduli "fire" per ridurre al minimo il numero di parametri mantenendo al contempo una forte accuratezza.
- MobileNet di Howard et al. (2017), d'altra parte, impiega le suddette convoluzioni separabili in profondità per ridurre sia il calcolo che le dimensioni del modello.
- EfficientNet di Tan e Le (2023) adotta un approccio diverso ottimizzando il ridimensionamento della rete (ovvero variando la profondità, la larghezza e la risoluzione di una rete) e il ridimensionamento composto, una variazione più sfumata del ridimensionamento della rete, per ottenere prestazioni superiori con meno parametri.

Questi modelli sono essenziali nel contesto dell'edge computing in cui la limitazione di potenza di elaborazione e di memoria richiede modelli leggeri ma efficaci in grado di eseguire in modo efficiente attività quali il riconoscimento delle immagini, il rilevamento di oggetti e altro ancora. I loro principi di progettazione mostrano l'importanza di un'architettura di modelli intenzionalmente personalizzata per l'edge computing, in cui prestazioni ed efficienza devono rientrare nei vincoli.

9.2.3.3 Semplificazione della Ricerca di Architetture di Modelli

Infine, per affrontare la sfida di trovare architetture di modelli efficienti che siano compatibili con i dispositivi edge, i ricercatori hanno sviluppato pipeline sistematizzate che semplificano la ricerca di progetti performanti. Due framework degni di nota in questo spazio sono TinyNAS di J. Lin et al. (2020) e MorphNet di Gordon et al. (2018), che automatizzano il processo di ottimizzazione delle architetture di reti neurali per l'implementazione edge.

TinyNAS è un innovativo framework di ricerca di architetture neurali introdotto nel documento MCUNet, progettato per scoprire in modo efficiente architetture di reti neurali leggere per dispositivi edge con risorse computazionali limitate. Sfruttando l'apprendimento per rinforzo e uno spazio di ricerca compatto di micromoduli neurali, TinyNAS ottimizza sia l'accuratezza che la latenza, consentendo l'implementazione di modelli di deep learning su microcontrollori, dispositivi IoT e altre piattaforme con risorse limitate. Nello specifico, TinyNAS, in combinazione con un ottimizzatore di rete, TinyEngine, genera diversi spazi di ricerca ridimensionando la risoluzione di input e la larghezza del modello, poi raccoglie la distribuzione FLOP di calcolo delle reti soddisfacenti all'interno dello spazio di ricerca per valutarne la priorità. TinyNAS si basa sul presupposto che uno spazio di ricerca che ospita FLOP più elevati con vincoli di memoria possa produrre modelli di accuratezza più elevata, cosa che gli autori hanno verificato in pratica nel loro lavoro. In termini di prestazioni empiriche, TinyEngine ha ridotto l'utilizzo di memoria di picco dei modelli di circa 3.4 volte e ha accelerato l'inferenza da 1.7 a 3.3 volte rispetto a TFLite e a CMSIS-NN.

Analogamente, MorphNet è un framework di ottimizzazione delle reti neurali progettato per rimodellare e trasformare automaticamente l'architettura delle reti neurali profonde, ottimizzandole per requisiti di distribuzione specifici. Ciò avviene in due fasi: in primo luogo, sfrutta un set di operazioni di morphing della rete personalizzabili, come l'ampliamento o l'approfondimento dei layer, per regolare dinamicamente la struttura della rete. Queste operazioni consentono alla rete di adattarsi a vari vincoli computazionali, tra cui dimensioni del modello, latenza e obiettivi di accuratezza, che sono estremamente diffusi nell'utilizzo dell'edge computing. Nella seconda fase, MorphNet utilizza un approccio basato sull'apprendimento di rinforzo per cercare la permutazione ottimale delle operazioni di

morphing, bilanciando efficacemente il compromesso tra dimensioni del modello e prestazioni. Questo metodo innovativo consente ai professionisti del deep learning di adattare automaticamente le architetture delle reti neurali a requisiti hardware e applicativi specifici, garantendo un'implementazione efficiente ed efficace su diverse piattaforme. TinyNAS e MorphNet rappresentano alcuni dei numerosi progressi significativi nel campo dell'ottimizzazione sistematica delle reti neurali, consentendo di scegliere e generare sistematicamente architetture per adattarsi perfettamente ai vincoli del problema.

Esercizio 9.3: Modelli Progettati per l'Edge

Si Immagini di costruire un piccolo robot in grado di identificare diversi fiori. Deve essere intelligente, ma anche piccolo ed efficiente dal punto di vista energetico! Nel mondo dell'"Edge-Aware Model Design", abbiamo appreso tecniche come le convoluzioni separabili in base alla profondità e architetture come SqueezeNet, MobileNet ed EfficientNet, tutte progettate per concentrare l'intelligenza in modelli compatti. Ora, vediamo queste idee in azione con alcuni xColab:

SqueezeNet in Action: Forse piacerebbe un Colab che mostra come addestrare un modello SqueezeNet su un set di dati di immagini di fiori. Ciò dimostrerebbe le sue piccole dimensioni e come impara a riconoscere i pattern nonostante la sua efficienza.

Open in Colab

MobileNet Exploration: Ci si è mai chiesto se quei piccoli modelli di immagini sono buoni quanto quelli grandi? Scopriamolo! In questo Colab, mettiamo a confronto MobileNet, il campione dei pesi leggeri, con un modello di classificazione delle immagini classico. Li faremo gareggiare per la velocità, misureremo le loro esigenze di memoria e vedremo chi vincerà per accuratezza. Preparatevi per una battaglia di cervelli di immagini!

Open in Colab

9.3 Rappresentazione Numerica Efficiente

La rappresentazione numerica implica una miriade di considerazioni, tra cui, ma non solo, la precisione dei numeri, i loro formati di codifica e le operazioni aritmetiche facilitate. Implica invariabilmente una vasta gamma di diversi compromessi, in cui i professionisti sono incaricati di destreggiarsi tra accuratezza numerica ed efficienza computazionale. Ad esempio, mentre i numeri a bassa precisione possono offrire il fascino di un utilizzo di memoria ridotto e calcoli accelerati, presentano contemporaneamente sfide relative alla stabilità numerica e al potenziale degrado dell'accuratezza del modello.

9.3.0.1 Motivazione

Emerge l'imperativo per una rappresentazione numerica efficiente, in particolare perché l'ottimizzazione efficiente del modello da sola non è sufficiente quando si adattano i modelli per l'implementazione su dispositivi edge a bassa potenza che operano con vincoli rigorosi. Oltre a ridurre al minimo le richieste di memoria, l'enorme potenziale di una rappresentazione numerica efficiente risiede, ma non è limitato a, queste modalità fondamentali. Riducendo l'intensità computazionale, la matematica efficiente può amplificare la velocità computazionale, consentendo di elaborare modelli più complessi su dispositivi a bassa potenza. Ridurre la precisione in bit di pesi e attivazioni su modelli fortemente sovra-parametrizzati consente la condensazione delle dimensioni del modello per dispositivi edge senza danneggiare significativamente l'accuratezza predittiva del modello. Con l'onnipresenza delle reti neurali nei modelli, la matematica efficiente ha un vantaggio unico nello sfruttare la struttura a layer delle NN per variare la precisione numerica

tra i layer, riducendo al minimo la precisione nei layer resistenti e preservando una maggiore precisione in quelli sensibili.
In questa sezione, approfondiremo il modo in cui i professionisti possono sfruttare i principi della progettazione congiunta hardware-software ai livelli più bassi di un modello per facilitare la compatibilità con i dispositivi edge. Iniziando con un'introduzione ai numeri, esamineremo le sue implicazioni per la memoria del dispositivo e la complessità computazionale. Successivamente, intraprenderemo una discussione sui compromessi implicati nell'adozione di questa strategia, seguita da un'analisi approfondita di un metodo fondamentale della matematica efficiente: la quantizzazione.

9.3.1 Le Basi

9.3.1.1 I Tipi

I dati numerici, il fondamento su cui si basano i modelli di apprendimento automatico, si manifestano in due forme principali. Si tratta di numeri interi e numeri in virgola mobile.

Numeri Interi: Numeri interi, privi di componenti frazionarie, (ad esempio, -3, 0, 42) sono fondamentali negli scenari che richiedono valori discreti. Ad esempio, in ML, le etichette di classe in un'attività di classificazione potrebbero essere rappresentate come numeri interi, dove "gatto", "cane" e "uccello" potrebbero essere codificati rispettivamente come 0, 1 e 2.

Numeri in virgola mobile: Comprendendo numeri reali, (ad esempio, -3.14, 0.01, 2.71828) consentono la rappresentazione di valori con componenti frazionarie. Nei parametri del modello ML, i pesi potrebbero essere inizializzati con piccoli valori a virgola mobile, ad esempio 0.001 o -0.045, per avviare il processo di training. Attualmente, ci sono 4 popolari formati di precisione discussi di seguito.

Larghezze di bit variabili: Oltre alle larghezze standard, sono in corso ricerche su numeri con larghezze di bit estremamente basse, persino fino a rappresentazioni binarie o ternarie. Le operazioni con larghezza di bit estremamente ridotta possono offrire accelerazioni significative e ridurre ulteriormente il consumo energetico. Sebbene permangano dei problemi nel mantenere l'accuratezza del modello con una quantizzazione così drastica, si continuano a fare progressi in quest'area.

9.3.1.2 Precisione

La precisione, che delinea l'esattezza con cui un numero è rappresentato, si biforca tipicamente in singola, doppia, mezza e negli ultimi anni sono emerse numerose altre precisioni per supportare meglio e in modo efficiente le attività di apprendimento automatico sull'hardware sottostante.

Doppia precisione (Float64): Allocando 64 bit, la doppia precisione (ad esempio, 3.141592653589793) fornisce una precisione elevata, sebbene richieda più memoria e più risorse di calcolo. Nei calcoli scientifici, dove la precisione è fondamentale, variabili come π potrebbero essere rappresentate con Float64.

Singola precisione (Float32): Con 32 bit a disposizione, la singola precisione (ad esempio, 3.1415927) raggiunge un equilibrio tra precisione numerica e risparmio della memoria. In ML, Float32 potrebbe essere impiegato per memorizzare i pesi durante l'addestramento per mantenere un livello ragionevole di precisione.

Half Precision (Float16): Limitata a 16 bit, la half precision (ad esempio, 3.14) riduce l'utilizzo della memoria e può velocizzare i calcoli, sebbene sacrifichi l'accuratezza e

l'intervallo numerico. In ML, specialmente durante l'inferenza su dispositivi con risorse limitate, Float16 potrebbe essere utilizzato per ridurre l'impronta di memoria del modello.

Bfloat16: Brain Floating-Point Format o Bfloat16, impiega anche 16 bit ma li alloca in modo diverso rispetto a FP16: 1 bit per il segno, 8 bit per l'esponente (che si traduce nello stesso intervallo numerico di float32) e 7 bit per la frazione. Questo formato, sviluppato da Google, dà priorità a un intervallo di esponenti più ampio rispetto alla precisione, rendendolo particolarmente utile nelle applicazioni di apprendimento profondo in cui l'intervallo dinamico è cruciale.
Figura 9.13 illustra le differenze tra i tre formati a virgola mobile: Float32, Float16 e BFloat16.

Figura 9.13: Tre formati a virgola mobile.

Intero: Le rappresentazioni di numeri interi sono realizzate utilizzando 8, 4 e 2 bit. Vengono spesso utilizzati durante la fase di inferenza delle reti neurali, in cui i pesi e le attivazioni del modello sono quantizzati a queste precisioni inferiori. Le rappresentazioni intere sono deterministiche e offrono notevoli vantaggi in termini di velocità e memoria rispetto alle rappresentazioni in virgola mobile. Per molte attività di inferenza, in particolare su dispositivi edge, la leggera perdita di accuratezza dovuta alla quantizzazione è spesso accettabile dati i guadagni di efficienza. Una forma estrema di numeri interi è per le reti neurali binarie (BNN), in cui pesi e attivazioni sono vincolati a uno di due valori: +1 o -1.
È possibile fare riferimento a Sezione 8.6.1 per una tabella di confronto tra i compromessi dei diversi tipi numerici.

9.3.1.3 Codifica e Archiviazione Numerica

La codifica numerica, l'arte di trasformare i numeri in un formato utilizzabile dal computer e la loro successiva memorizzazione sono fondamentali per l'efficienza computazionale. Ad esempio, i numeri in virgola mobile potrebbero essere codificati utilizzando lo standard IEEE 754, che ripartisce i bit tra i componenti segno, esponente e frazione, consentendo così la rappresentazione di una vasta gamma di valori con un singolo formato. Esistono alcuni nuovi formati in virgola mobile IEEE che sono stati definiti specificamente per i carichi di lavoro AI:

- bfloat16- Un formato in virgola mobile a 16 bit introdotto da Google. Ha 8 bit per esponente, 7 bit per mantissa e 1 bit per segno. Offre un compromesso di precisione ridotto tra float a 32 bit e interi a 8 bit. Supportato su molti acceleratori hardware.

- posit - Un formato configurabile che può rappresentare diversi livelli di precisione in base ai bit esponente. È più efficiente dei numeri binari in virgola mobile IEEE 754. Ha una gamma dinamica e una precisione regolabili.
- Flexpoint - Un formato introdotto da Intel che può regolare dinamicamente la precisione tra livelli o all'interno di un layer. Consente di adattare la precisione all'accuratezza e ai requisiti hardware.
- BF16ALT - Un formato a 16 bit proposto da ARM come alternativa a bfloat16. Utilizza un bit aggiuntivo nell'esponente per evitare overflow/underflow.
- TF32 - Introdotto da Nvidia per le GPU Ampere. Utilizza 10 bit per l'esponente invece di 8 bit come FP32. Migliora le prestazioni di training del modello mantenendo l'accuratezza.
- FP8 - Formato a virgola mobile a 8 bit che mantiene 6 bit per la mantissa e 2 bit per l'esponente. Consente una gamma dinamica migliore rispetto agli interi.

Gli obiettivi principali di questi nuovi formati sono di fornire alternative di precisione inferiore ai float a 32 bit per una migliore efficienza computazionale e prestazioni sugli acceleratori AI, mantenendo al contempo l'accuratezza del modello. Offrono diversi compromessi in termini di precisione, portata e costo/complessità di implementazione.

9.3.2 Vantaggi dell'Efficienza

Come visto in Sezione 8.6.2, l'efficienza numerica è importante per i carichi di lavoro di apprendimento automatico per una serie di motivi. L'efficienza numerica non riguarda solo la riduzione della larghezza di bit dei numeri, ma anche la comprensione dei compromessi tra accuratezza ed efficienza. Man mano che i modelli di apprendimento automatico diventano più pervasivi, soprattutto in ambienti reali con risorse limitate, l'attenzione su una numerica efficiente continuerà a crescere. Selezionando e sfruttando attentamente la precisione numerica appropriata, è possibile ottenere prestazioni di modello robuste ottimizzando al contempo velocità, memoria ed energia.

9.3.3 Sfumature della Rappresentazione Numerica

Ci sono diverse sfumature con le rappresentazioni numeriche per ML che richiedono di avere una comprensione sia degli aspetti teorici che pratici della rappresentazione numerica, nonché una profonda consapevolezza dei requisiti e dei vincoli specifici del dominio applicativo.

9.3.3.1 Utilizzo della Memoria

L'impronta di memoria dei modelli ML, in particolare quelli di notevole complessità e profondità, può essere sostanziale, ponendo quindi una sfida significativa sia nelle fasi di training che di deployment. Ad esempio, una rete neurale profonda con 100 milioni di parametri, rappresentata utilizzando Float32 (32 bit o 4 byte per parametro), richiederebbe circa 400 MB di memoria solo per l'archiviazione dei pesi del modello. Ciò non tiene conto dei requisiti di memoria aggiuntivi durante il training per l'archiviazione di gradienti, stati dell'ottimizzatore e cache di passaggio forward [in avanti], che possono amplificare ulteriormente l'utilizzo della memoria, potenzialmente mettendo a dura prova le risorse su determinati hardware, in particolare dispositivi edge con capacità di memoria limitata.
La scelta della rappresentazione numerica ha un impatto ulteriore sull'utilizzo della memoria e sull'efficienza computazionale. Ad esempio, l'utilizzo di Float64 per i pesi del modello raddoppierebbe i requisiti di memoria rispetto a Float32 e potrebbe potenzialmente aumentare anche il tempo di elaborazione. Per una matrice di peso con dimensioni [1000, 1000], Float64 consumerebbe circa 8 MB di memoria, mentre Float32 la ridurrebbe a circa 4

MB. Pertanto, la selezione di un formato numerico appropriato è fondamentale per ottimizzare sia la memoria che l'efficienza computazionale.

9.3.3.2 Complessità Computazionale

La precisione numerica ha un impatto diretto sulla complessità computazionale, influenzando il tempo e le risorse necessarie per eseguire operazioni aritmetiche. Ad esempio, le operazioni che utilizzano Float64 generalmente consumano più risorse computazionali rispetto alle loro controparti Float32 o Float16 (vedere Figura 9.14). Nel regno del ML, dove i modelli potrebbero dover elaborare milioni di operazioni (ad esempio, moltiplicazioni e addizioni in operazioni di matrice durante passaggi in forward e backward), anche piccole differenze nella complessità computazionale per operazione possono aggregarsi in un impatto sostanziale sui tempi di training e inferenza. Come mostrato in Figura 9.15, i modelli quantizzati possono essere molte volte più veloci delle loro versioni non-quantizzate.

Integer	
Add	
8 bit	0.03pJ
32 bit	0.1pJ
Mult	
8 bit	0.2pJ
32 bit	3.1pJ

FP	
FAdd	
16 bit	0.4pJ
32 bit	0.9pJ
FMult	
16 bit	1.1pJ
32 bit	3.7pJ

Memory	
Cache	(64bit)
8KB	10pJ
32KB	20pJ
1MB	100pJ
DRAM	1.3-2.6nJ

Figura 9.14: Utilizzo di energia da parte di operazioni quantizzate. Fonte: Mark Horowitz, Stanford University.

Figura 9.15: Velocità di tre diversi modelli in forma normale e quantizzata.

Oltre ai tempi di esecuzione puri, c'è anche una preoccupazione per l'efficienza energetica. Non tutti i calcoli numerici sono creati uguali dal punto di vista dell'hardware sottostante. Alcune operazioni numeriche sono più efficienti dal punto di vista energetico di altre. Ad esempio, Figura 9.16 di seguito mostra che l'addizione di interi è molto più efficiente dal punto di vista energetico della moltiplicazione di interi.

Operation	Energy (pJ)
Integer ADD (8b)	0.03
Integer ADD (16b)	0.05
Integer ADD (32b)	0.1
Integer MULT (8b)	0.2
Integer MULT (32b)	3.1
8KB SRAM Read (32b)	5
32KB SRAM Read (32b)	10
1MB SRAM Read (32b)	50

Figura 9.16: Utilizzo di energia da parte di operazioni quantizzate. Fonte: Isscc (2014).

9.3.3.3 Compatibilità Hardware

Garantire la compatibilità e le prestazioni ottimizzate su diverse piattaforme hardware è un'altra sfida nella rappresentazione numerica. Hardware diversi, come CPU, GPU, TPU e FPGA, hanno capacità e ottimizzazioni diverse per gestire diverse precisioni numeriche. Ad esempio, alcune GPU potrebbero essere ottimizzate per i calcoli Float32, mentre altre potrebbero fornire accelerazioni per Float16. Sviluppare e ottimizzare modelli ML in grado di sfruttare le capacità numeriche specifiche di hardware diversi, garantendo al contempo che il modello mantenga la sua accuratezza e robustezza, richiede un'attenta considerazione e potenzialmente ulteriori sforzi di sviluppo e test.

9.3.3.4 Compromessi di Precisione e Accuratezza

Il compromesso tra precisione numerica e accuratezza del modello è una sfida "sfumata" nella rappresentazione numerica. L'utilizzo di numeri a bassa precisione, come Float16, potrebbe risparmiare memoria e velocizzare i calcoli, ma può anche introdurre problemi come errore di quantizzazione e intervallo numerico ridotto. Ad esempio, addestrare un modello con Float16 potrebbe introdurre problemi nella rappresentazione di valori di gradiente molto piccoli, potenzialmente influenzando la convergenza e la stabilità del processo di addestramento. Inoltre, in alcune applicazioni, come simulazioni scientifiche o calcoli finanziari, in cui l'elevata precisione è fondamentale, l'uso di numeri a bassa precisione potrebbe non essere consentito a causa del rischio di accumulare errori significativi.

9.3.3.5 Esempi di Compromessi

Per comprendere e apprezzare le sfumature, prendiamo in considerazione alcuni esempi di casi d'uso. Attraverso questi, ci renderemo conto che la scelta della rappresentazione numerica non è semplicemente una decisione tecnica, ma strategica, che influenza l'acume predittivo del modello, le sue esigenze computazionali e la sua implementabilità in diversi ambienti computazionali. In questa sezione esamineremo un paio di esempi per comprendere meglio i compromessi con i numeri e come si collegano al mondo reale.

9.3.3.5.1 Veicoli Autonomi

Nel dominio dei veicoli autonomi, i modelli ML vengono impiegati per interpretare i dati dei sensori e prendere decisioni in tempo reale. I modelli devono elaborare dati ad alta

dimensionalità da vari sensori (ad esempio, LiDAR, telecamere, radar) ed eseguire numerosi calcoli entro un intervallo di tempo limitato per garantire un funzionamento sicuro e reattivo del veicolo. Quindi i compromessi qui includerebbero:

- Utilizzo della Memoria: L'archiviazione e l'elaborazione di dati dei sensori ad alta risoluzione, specialmente in formati a virgola mobile, possono consumare una quantità di memoria sostanziale.
- Complessità Computazionale: L'elaborazione in tempo reale richiede calcoli efficienti, in cui numeri di precisione più elevata potrebbero impedire l'esecuzione tempestiva delle azioni di controllo.

9.3.3.5.2 Applicazioni Sanitarie Mobili

Le applicazioni sanitarie mobili spesso utilizzano modelli ML per attività come il riconoscimento delle attività, il monitoraggio della salute o l'analisi predittiva, operando nell'ambiente con risorse limitate dei dispositivi mobili. I compromessi in questo caso includerebbero:

- Compromessi di Precisione e Accuratezza: L'impiego di numeri a bassa precisione per conservare risorse potrebbe influire sull'accuratezza delle previsioni sanitarie o delle rilevazioni di anomalie, il che potrebbe avere implicazioni significative per la salute e la sicurezza degli utenti.
- Compatibilità Hardware: I modelli devono essere ottimizzati per diversi hardware mobili, garantendo un funzionamento efficiente su un'ampia gamma di dispositivi con diverse capacità di calcolo numerico.

9.3.3.5.3 Sistemi di Trading ad Alta Frequenza (HFT)

I sistemi HFT sfruttano i modelli ML per prendere decisioni di trading rapide basate su dati di mercato in tempo reale. Questi sistemi richiedono risposte a bassissima latenza per capitalizzare le opportunità di trading di breve durata.

- Complessità Computazionale: I modelli devono elaborare e analizzare vasti flussi di dati di mercato con una latenza minima, dove anche lievi ritardi, potenzialmente introdotti da numeri a precisione più elevata, possono comportare opportunità perse.
- Compromessi di Precisione e Accuratezza: I calcoli finanziari spesso richiedono un'elevata precisione numerica per garantire valutazioni accurate dei prezzi e dei rischi, ponendo sfide nel bilanciamento tra efficienza computazionale e accuratezza numerica.

9.3.3.5.4 Sistemi di Sorveglianza Basati su Edge

I sistemi di sorveglianza distribuiti su dispositivi edge, come le telecamere di sicurezza, utilizzano modelli ML per attività come rilevamento di oggetti, riconoscimento di attività e rilevamento di anomalie, spesso operando con vincoli di risorse rigorosi.

- Utilizzo della Memoria: L'archiviazione di modelli pre-addestrati e l'elaborazione di feed video in tempo reale richiedono un utilizzo efficiente della memoria, il che può essere impegnativo con numeri ad alta precisione.
- Compatibilità Hardware: Garantire che i modelli possano funzionare in modo efficiente su dispositivi edge con diverse capacità hardware e ottimizzazioni per diverse precisioni numeriche è fondamentale per una distribuzione diffusa.

9.3.3.5.5 Simulazioni Scientifiche

I modelli ML vengono sempre più utilizzati nelle simulazioni scientifiche, come la modellazione climatica o le simulazioni di dinamica molecolare, per migliorare le capacità predittive e ridurre le richieste di calcolo.

- Compromessi di Precisione e Accuratezza: Le simulazioni scientifiche spesso richiedono un'elevata precisione numerica per garantire risultati accurati e affidabili, il che può entrare in conflitto con il desiderio di ridurre le richieste di calcolo tramite numeri a bassa precisione.
- Complessità Computazionale: I modelli devono gestire ed elaborare dati di simulazione complessi e ad alta dimensionalità in modo efficiente per garantire risultati tempestivi e consentire simulazioni su larga scala o di lunga durata.

Questi esempi illustrano diversi scenari in cui le sfide della rappresentazione numerica nei modelli ML sono palesemente manifestate. Ogni sistema presenta un set unico di requisiti e vincoli, che richiedono strategie e soluzioni personalizzate per affrontare i problemi dell'utilizzo della memoria, della complessità computazionale, dei compromessi tra precisione e accuratezza e della compatibilità hardware.

9.3.4 Quantizzazione

La quantizzazione è prevalente in vari domini scientifici e tecnologici e comporta essenzialmente la mappatura o la limitazione di un set o intervallo continuo in una controparte discreta per ridurre al minimo il numero di bit richiesti.

9.3.4.1 Analisi Iniziale

Iniziamo la nostra incursione nella quantizzazione con una breve analisi di un importante utilizzo della quantizzazione.
Nel signal processing [elaborazione del segnale], l'onda sinusoidale continua (mostrata in Figura 9.17) può essere quantizzata in valori discreti tramite un processo noto come campionamento. Questo è un concetto fondamentale nell'elaborazione del segnale digitale ed è cruciale per convertire segnali analogici (come l'onda sinusoidale continua) in una forma digitale che possa essere elaborata dai computer. L'onda sinusoidale è un esempio prevalente grazie alla sua natura periodica e regolare, il che la rende uno strumento utile per spiegare concetti come frequenza, ampiezza, fase e, naturalmente, quantizzazione.

Figura 9.17: Onda Sinusoidale.

Nella versione quantizzata mostrata in Figura 9.18, l'onda sinusoidale continua (Figura 9.17) viene campionata a intervalli regolari (in questo caso, ogni $\frac{\pi}{4}$ radianti) e solo questi valori campionati vengono rappresentati nella versione digitale del segnale. Le linee graduali tra i punti mostrano un modo per rappresentare il segnale quantizzato in una forma costante a tratti. Questo è un esempio semplificato di come funziona la conversione analogico-digitale,

in cui un segnale continuo viene mappato su un set discreto di valori, consentendone la rappresentazione e l'elaborazione digitale.

Figura 9.18: Onda Sinusoidale Quantizzata.

Tornando al contesto del Machine Learning (ML), la quantizzazione si riferisce al processo di limitazione dei possibili valori che i parametri numerici (come pesi e bias) possono assumere in un set discreto, riducendo così la precisione dei parametri e, di conseguenza, l'ingombro di memoria del modello. Se implementata correttamente, la quantizzazione può ridurre le dimensioni del modello fino a 4 volte e migliorare la latenza e la produttività dell'inferenza fino a 2-3 volte. Figura 9.19 illustra l'impatto che la quantizzazione ha sulle dimensioni di modelli diversi: ad esempio, un modello di classificazione delle immagini come ResNet-v2 può essere compresso da 180 MB a 45 MB con quantizzazione a 8 bit. In genere, la perdita di accuratezza del modello è inferiore all'1% con una quantizzazione ben fatta. L'accuratezza può spesso essere recuperata riaddestrando il modello quantizzato con tecniche di addestramento consapevoli della quantizzazione. Pertanto, questa tecnica è emersa come molto importante nell'implementazione di modelli ML in ambienti con risorse limitate, come dispositivi mobili, dispositivi IoT e piattaforme di edge computing, dove le risorse computazionali (memoria e potenza di elaborazione) sono limitate.

Figura 9.19: Effetto della quantizzazione sulle dimensioni del modello. Fonte: HarvardX.

Esistono diverse dimensioni della quantizzazione, come uniformità, stocasticità (o determinismo), simmetria, granularità (tra layer/canali/gruppi o persino all'interno dei canali), considerazioni sulla calibrazione dell'intervallo (statico o dinamico) e metodi di messa a punto (QAT, PTQ, ZSQ). Esaminiamo questi di seguito.

9.3.5 I Tipi

9.3.5.1 Quantizzazione Uniforme

La quantizzazione uniforme implica la mappatura di valori continui o ad alta precisione su una rappresentazione a precisione inferiore utilizzando una scala uniforme. Ciò significa che l'intervallo tra ogni possibile valore quantizzato è coerente. Ad esempio, se i pesi di un layer di rete neurale sono quantizzati su numeri interi a 8 bit (valori tra 0 e 255), un peso con un valore in virgola mobile di 0.56 potrebbe essere mappato su un valore intero di 143, presupponendo una mappatura lineare tra le scale originale e quantizzata. Grazie all'uso di pipeline matematiche intere o a virgola fissa, questa forma di quantizzazione consente il calcolo sul dominio quantizzato senza la necessità di dequantizzare in anticipo.
Il processo per implementare la quantizzazione uniforme inizia con la scelta di un intervallo di numeri reali da quantizzare. Il passaggio successivo consiste nel selezionare una funzione di quantizzazione e mappare i valori reali sugli interi rappresentabili dalla larghezza di bit della rappresentazione quantizzata. Ad esempio, una scelta popolare per una funzione di quantizzazione è:

$$Q(r) = Int(r/S) - Z$$

dove Q è l'operatore di quantizzazione, r è un input a valore reale (nel nostro caso, un'attivazione o un peso), S è un fattore di scala a valore reale e Z è un punto zero intero. La funzione `Int` mappa un valore reale in un valore intero tramite un'operazione di arrotondamento. Tramite questa funzione, abbiamo mappato in modo efficace i valori reali r in alcuni valori interi, ottenendo livelli quantizzati uniformemente distanziati.

Quando i professionisti hanno la necessità di recuperare i valori originali di precisione più elevata, i valori reali r possono essere recuperati dai valori quantizzati tramite un'operazione

nota come **dequantizzazione**. Nell'esempio sopra, ciò significherebbe eseguire la seguente operazione sul nostro valore quantizzato:

$$\bar{r} = S(Q(r) + Z)$$

Come discusso, una certa precisione nel valore reale viene persa dalla quantizzazione. In questo caso, il valore recuperato $\bar{r}$ non corrisponderà esattamente a r a causa dell'operazione di arrotondamento. Questo è un importante compromesso da notare; tuttavia, in molti utilizzi riusciti della quantizzazione, la perdita di precisione può essere trascurabile e l'accuratezza del test rimane elevata. Nonostante ciò, la quantizzazione uniforme continua a essere la scelta di fatto attuale per la sua semplicità e l'efficiente mappatura all'hardware.

9.3.5.2 Quantizzazione Non-Uniforme

La quantizzazione non uniforme, d'altro canto, non mantiene un intervallo coerente tra i valori quantizzati. Questo approccio potrebbe essere utilizzato per allocare più possibili valori discreti in regioni in cui i valori dei parametri sono più densamente popolati, preservando così maggiori dettagli dove sono più necessari. Ad esempio, nelle distribuzioni a campana di pesi con lunghe code, un set di pesi in un modello si trova prevalentemente all'interno di un certo intervallo; quindi, più livelli di quantizzazione potrebbero essere assegnati a tale intervallo per preservare dettagli più fini, consentendoci di acquisire meglio le informazioni. Tuttavia, una delle principali debolezze della quantizzazione non uniforme è che richiede la dequantizzazione prima di calcoli di precisione più elevata a causa della sua non uniformità, limitando la sua capacità di accelerare il calcolo rispetto alla quantizzazione uniforme.

In genere, una quantizzazione non uniforme basata su regole utilizza una distribuzione logaritmica di passaggi e livelli esponenzialmente crescenti anziché linearmente. Un altra tipologia popolare risiede nella quantizzazione basata su codice binario in cui i vettori di numeri reali vengono quantizzati in vettori binari con un fattore di scala. In particolare, non esiste una soluzione in forma chiusa per minimizzare gli errori tra il valore reale e il valore non uniformemente quantizzato, quindi la maggior parte delle quantizzazioni in questo campo si basa su soluzioni euristiche. Ad esempio, un lavoro recente di C. Xu et al. (2018) formula la quantizzazione non uniforme come un problema di ottimizzazione in cui i passaggi/livelli di quantizzazione nel quantizzatore Q vengono regolati per ridurre al minimo la differenza tra il tensore originale e la controparte quantizzata.

$$\min_{Q} \| \quad |Q(r) - r| \quad |^2$$

Inoltre, i quantizzatori addestrabili lo possono essere congiuntamente con parametri di modello e i passaggi/livelli di quantizzazione sono generalmente addestrati con ottimizzazione iterativa o discesa del gradiente. Inoltre, il clustering è stato utilizzato per alleviare la perdita di informazioni dalla quantizzazione. Sebbene in grado di catturare livelli di dettaglio più elevati, gli schemi di quantizzazione non uniformi possono essere difficili da implementare in modo efficiente su hardware di calcolo generale, rendendoli meno preferiti ai metodi che utilizzano la quantizzazione uniforme.

Figura 9.20: Uniformità della Quantizzazione. Fonte: Gholami et al. (2021).

9.3.5.3 Quantizzazione Stocastica

A differenza dei due approcci precedenti che generano mappature deterministiche, c'è un po' di lavoro che esplora l'idea della quantizzazione stocastica per l'addestramento consapevole della quantizzazione e l'addestramento a precisione ridotta. Questo approccio mappa numeri fluttuanti verso l'alto o verso il basso con una probabilità associata alla grandezza dell'aggiornamento del peso. La speranza generata dall'intuizione di alto livello è che un tale approccio probabilistico possa consentire a una rete neurale di esplorare di più, rispetto alla quantizzazione deterministica. Presumibilmente, abilitare un arrotondamento stocastico potrebbe consentire alle reti neurali di sfuggire agli ottimi locali, aggiornando così i propri parametri. Di seguito sono riportati due esempi di funzioni di mappatura stocastica:

$$\mathrm{Int}(x) = \begin{cases} \lfloor x \rfloor & \text{with probability } \lceil x \rceil - x, \\ \lceil x \rceil & \text{with probability } x - \lfloor x \rfloor. \end{cases}$$

$$\mathrm{Binary}(x) = \begin{cases} -1 & \text{with probability } 1 - \sigma(x), \\ +1 & \text{with probability } \sigma(x), \end{cases}$$

Figura 9.21: Funzioni di quantizzazione Intera e Binaria.

9.3.5.4 Quantizzazione "Zero Shot"

La quantizzazione Zero-shot si riferisce al processo di conversione di un modello di deep learning a precisione completa direttamente in un modello quantizzato a bassa precisione senza la necessità di alcun riaddestramento o messa a punto sul modello quantizzato. Il vantaggio principale di questo approccio è la sua efficienza, in quanto elimina il processo, spesso dispendioso in termini di tempo e risorse, del riaddestramento post-quantizzazione. Sfruttando tecniche che anticipano e riducono al minimo gli errori di quantizzazione, la quantizzazione zero-shot mantiene l'accuratezza originale del modello anche dopo averne ridotto la precisione numerica. È particolarmente utile per i provider di "Machine Learning as a Service (MLaaS)" che mirano ad accelerare la distribuzione dei carichi di lavoro dei propri clienti senza dover accedere ai loro set di dati.

9.3.6 Calibrazione

La calibrazione è il processo di selezione dell'intervallo di clipping [ritaglio] più efficace $[\alpha, \beta]$ per pesi e attivazioni da quantizzare. Ad esempio, si consideri la quantizzazione delle

attivazioni che originariamente hanno un intervallo in virgola mobile tra -6 e 6 a interi a 8 bit. Prendere solo i valori minimi e massimi possibili di interi a 8 bit (da -128 a 127) come intervallo di quantizzazione, potrebbe non essere il più efficace. Invece, la calibrazione implicherebbe il passaggio di un set di dati rappresentativo e quindi l'utilizzo di questo intervallo osservato per la quantizzazione.
Esistono molti metodi di calibrazione, ma alcuni comunemente utilizzati includono:

- Max: Utilizza il valore assoluto massimo visualizzato durante la calibrazione. Tuttavia, questo metodo è suscettibile di dati anomali. Notare come in Figura 9.22, abbiamo un cluster anomalo intorno a 2.1, mentre il resto è raggruppato attorno a valori più piccoli.
- Entropia: Utilizza la divergenza KL per ridurre al minimo la perdita di informazioni tra i valori originali in virgola mobile e i valori che potrebbero essere rappresentati dal formato quantizzato. Questo è il metodo predefinito utilizzato da TensorRT.
- Percentile: Imposta l'intervallo su un percentile della distribuzione dei valori assoluti osservati durante la calibrazione. Ad esempio, una calibrazione del 99% taglierebbe l'1% dei valori di magnitudine più grandi.

Figura 9.22: Attivazioni di input nel layer 3 in ResNet50. Fonte: @Wu, Judd, e Isaev (2020).

È importante notare che la qualità della calibrazione può fare la differenza tra un modello quantizzato che conserva la maggior parte della sua accuratezza e uno che si degrada in modo significativo. Quindi, è un passaggio essenziale nel processo di quantizzazione. Quando si sceglie un intervallo di calibrazione, ci sono due tipi: simmetrico e asimmetrico.

9.3.6.1 Quantizzazione Simmetrica

La quantizzazione simmetrica mappa i valori reali su un intervallo di clipping simmetrico centrato su 0. Ciò comporta la scelta di un intervallo $[\alpha, \beta]$ dove $\alpha = -\beta$. Ad esempio, un intervallo simmetrico si baserebbe sui valori min/max dei valori reali in modo tale che:

$$\alpha = \beta = max\big(abs(r_{max}), abs(r_{min})\big)$$

Gli intervalli di clipping simmetrici sono i più ampiamente adottati nella pratica in quanto hanno il vantaggio di un'implementazione più semplice. In particolare, la mappatura da zero a zero nell'intervallo di clipping (talvolta chiamata "azzeramento del punto zero") può portare a una riduzione del costo computazionale durante l'inferenza (Wu, Judd, e Isaev 2020).

9.3.6.2 Quantizzazione Asimmetrica

La quantizzazione asimmetrica mappa i valori reali in un intervallo di clipping asimmetrico che non è necessariamente centrato sullo 0, come mostrato in Figura 9.23 a destra. Comporta la scelta di un intervallo $[\alpha, \beta]$ dove $\alpha \neq -\beta$. Ad esempio, selezionando un intervallo basato sui valori reali minimi e massimi, o dove $\alpha = r_{min}$ and $\beta = r_{max}$, si crea un intervallo asimmetrico. In genere, la quantizzazione asimmetrica produce intervalli di clipping più stretti rispetto a quella simmetrica, il che è importante quando i pesi e le attivazioni target sono sbilanciati, ad esempio, l'attivazione dopo la ReLU ha sempre valori non negativi. Nonostante produca intervalli di clipping più stretti, la quantizzazione asimmetrica è meno preferita di quella simmetrica in quanto non azzera sempre il valore dello zero reale.

Figura 9.23: (a)simmetria della Quantizzazione. Fonte: Gholami et al. (2021).

9.3.6.3 Granularità

Dopo aver deciso il tipo di intervallo di clipping, è essenziale restringerlo per consentire a un modello di mantenere la massima accuratezza possibile. Daremo un'occhiata alle reti neurali convoluzionali come nostro modo di esplorare metodi che ottimizzano la granularità degli intervalli di clipping per la quantizzazione. L'attivazione di input di un layer nella nostra CNN subisce una convoluzione con più filtri convoluzionali. Ogni filtro convoluzionale può possedere un intervallo di valori univoco. Si noti come in Figura 9.24 l'intervallo per il Filtro 1 sia molto più piccolo di quello per il Filtro 3. Di conseguenza, una caratteristica distintiva degli approcci di quantizzazione è la precisione con cui l'intervallo di clipping [α,β] viene determinato per i pesi.

Figura 9.24: Granularità di quantizzazione: intervalli variabili. Fonte: Gholami et al. (2021).

1. **Quantizzazione a Layer:** Questo approccio determina l'intervallo di clipping considerando tutti i pesi nei filtri convoluzionali di un layer. Quindi, lo stesso intervallo di clipping viene utilizzato per tutti i filtri convoluzionali. È il più semplice da implementare e, come tale, spesso si traduce in una precisione non ottimale a causa dell'ampia varietà di intervalli diversi tra i filtri. Ad esempio, un kernel convoluzionale con un intervallo di parametri più ristretto perde la sua risoluzione di quantizzazione a causa di un altro kernel nello stesso layer che ha un intervallo più ampio.
2. **Groupwise Quantization:** Questo approccio raggruppa diversi canali all'interno di un layer per calcolare l'intervallo di clipping. Questo metodo può essere utile quando la distribuzione dei parametri su una singola convoluzione/attivazione varia molto. In pratica, questo metodo è stato utile in Q-BERT (Shen et al. 2020) per quantizzare i modelli Transformer (Vaswani et al. 2017) costituiti da layer di attenzione completamente connessi. Lo svantaggio di questo approccio è il costo aggiuntivo di contabilizzazione di diversi fattori di scala.
3. **Channelwise Quantization:** Questo metodo popolare utilizza un intervallo fisso per ogni filtro convoluzionale che è indipendente dagli altri canali. Poiché a ogni canale viene assegnato un fattore di scala dedicato, questo metodo garantisce una risoluzione di quantizzazione più elevata e spesso si traduce in una maggiore accuratezza.
4. **Sub-channelwise Quantization:** Portando la quantizzazione canale per canale all'estremo, questo metodo determina l'intervallo di clipping rispetto a qualsiasi gruppo di parametri in una convoluzione o in un layer completamente connesso. Potrebbe comportare un overhead considerevole poiché è necessario tenere conto di diversi fattori di scala quando si elabora una singola convoluzione o un layer completamente connesso.

Tra questi, la quantizzazione canale per canale è lo standard corrente utilizzato per quantizzare i kernel convoluzionali, poiché consente la regolazione degli intervalli di clipping per ogni singolo kernel con overhead trascurabile.

9.3.6.4 Quantizzazione Statica e Dinamica

Dopo aver determinato il tipo e la granularità dell'intervallo di clipping, gli esperti devono decidere quando gli intervalli vengono determinati nei loro algoritmi di calibrazione dell'intervallo. Esistono due approcci per quantizzare le attivazioni: quantizzazione statica e quella dinamica.

La quantizzazione statica è l'approccio più frequentemente utilizzato. In questo, l'intervallo di clipping è precalcolato e statico durante l'inferenza. Non aggiunge alcun sovraccarico computazionale, ma, di conseguenza, comporta una minore accuratezza rispetto alla quantizzazione dinamica. Un metodo popolare per implementarlo è eseguire una serie di input di calibrazione per calcolare l'intervallo tipico di attivazioni (Jacob et al. 2018; Yao et al. 2021).

La quantizzazione dinamica è un approccio alternativo che calcola dinamicamente l'intervallo per ogni mappa di attivazione durante il runtime. L'approccio richiede calcoli in tempo reale che potrebbero avere un sovraccarico molto elevato. In questo modo, la quantizzazione dinamica spesso raggiunge la massima accuratezza poiché l'intervallo viene calcolato specificamente per ogni input.

Tra i due, il calcolo dell'intervallo in modo dinamico è solitamente molto costoso, quindi la maggior parte dei professionisti utilizzerà spesso la quantizzazione statica.

9.3.7 Tecniche

Quando si ottimizzano i modelli di apprendimento automatico per l'implementazione, vengono utilizzate varie tecniche di quantizzazione per bilanciare efficienza, accuratezza e adattabilità del modello. Ogni metodo (quantizzazione post-addestramento, addestramento consapevole della quantizzazione e quantizzazione dinamica) offre vantaggi e compromessi unici, che incidono su fattori quali complessità di implementazione, sovraccarico computazionale e ottimizzazione delle prestazioni.

Tabella 9.2 fornisce una panoramica di questi metodi di quantizzazione, evidenziandone i rispettivi punti di forza, limiti e compromessi. Ci addentreremo più a fondo in ciascuno di questi metodi perché sono ampiamente distribuiti e utilizzati in tutti i sistemi ML di scale molto diverse.

Tabella 9.2: Confronto tra quantizzazione post-training, training sensibile alla quantizzazione e quantizzazione dinamica.

Aspetto	Quantizzazione Post Training	Training Quantization-Aware	Quantization Dinamica
Pro			
Semplicità	✓	✗	✗
Preservazione della Precisione	✗	✓	✓
Adattabilità	✗	✗	✓
Prestazioni Ottimizzate	✗	✓	Potentialmente
Contro			
Degrado della Precisione	✓	✗	Potentialmente
Overhead Computazionale	✗	✓	✓
Complessità di Implementazione	✗	✓	✓
Compromessi			
Velocità vs. Precisione	✓	✗	✗
Precisione vs. Costo	✗	✓	✗
Adaptability vs. Overhead	✗	✗	✓

Post Training Quantization: La quantizzazione post-addestramento (PTQ) è una tecnica di quantizzazione in cui il modello viene quantizzato dopo essere stato addestrato. Il modello viene addestrato in virgola mobile e poi i pesi e le attivazioni vengono quantizzati come fase di post-elaborazione. Questo è l'approccio più semplice e non richiede l'accesso ai dati di addestramento. Diversamente la "Quantization-Aware Training (QAT), PTQ" imposta direttamente i parametri di quantizzazione del peso e dell'attivazione, rendendolo poco costoso e adatto a situazioni con dati limitati o non etichettati. Tuttavia, non riaggiustare i pesi dopo la quantizzazione, specialmente nella quantizzazione a bassa precisione, può portare a un comportamento molto diverso e quindi a una minore accuratezza. Per affrontare questo problema, sono state sviluppate tecniche come la correzione della distorsione, l'equalizzazione degli intervalli di peso e i metodi di arrotondamento adattivo. PTQ può essere

applicato anche in scenari zero-shot, in cui non sono disponibili dati di addestramento o di test. Questo metodo è stato reso ancora più efficiente per avvantaggiare modelli linguistici di grandi dimensioni che richiedono molta elaborazione e memoria. Di recente, è stata sviluppata SmoothQuant, una soluzione PTQ senza training, che preserva l'accuratezza ed è di uso generale che consente la quantizzazione di peso a 8 bit e attivazione a 8 bit per LLM, dimostrando un'accelerazione fino a 1.56x e una riduzione della memoria di 2x per LLM con una perdita trascurabile di accuratezza (Xiao et al. 2022).
In PTQ, un modello pre-addestrato subisce un processo di calibrazione, come mostrato in Figura 9.25. La calibrazione comporta l'utilizzo di un set di dati separato noto come dati di calibrazione, un sottoinsieme specifico dei dati di training riservato alla quantizzazione per aiutare a trovare gli intervalli di clipping e i fattori di scala appropriati.

Figura 9.25: Quantizzazione e Calibrazione Post-Training. Fonte: Gholami et al. (2021).

Quantization-Aware Training: L'addestramento consapevole della quantizzazione (QAT) è una messa a punto del modello PTQ. Il modello viene addestrato in modo consapevole della quantizzazione, consentendogli di adattarsi agli effetti della quantizzazione. Ciò produce una migliore accuratezza con l'inferenza quantizzata. La quantizzazione di un modello di rete neurale addestrato con metodi come PTQ introduce perturbazioni che possono deviare il modello dal suo punto di convergenza originale. Ad esempio, Krishnamoorthi ha dimostrato che anche con la quantizzazione per canale, reti come MobileNet non raggiungono la precisione di base con int8 PTQ e richiedono QAT (Krishnamoorthi 2018). Per risolvere questo problema, QAT riqualifica il modello con parametri quantizzati, utilizzando passaggi forward [avanti] e backward [indietro] in virgola mobile ma quantizzando i parametri dopo ogni aggiornamento del gradiente. La gestione dell'operatore di quantizzazione non differenziabile è fondamentale; un metodo ampiamente utilizzato è lo "Straight Through Estimator (STE)", che approssima l'operazione di arrotondamento come una funzione identità. Sebbene esistano altri metodi e varianti, STE rimane il più comunemente utilizzato per la sua efficacia pratica. In QAT, un modello pre-addestrato viene quantizzato e poi messo a punto utilizzando i dati di addestramento per regolare i parametri e recuperare il degrado della precisione, come mostrato in Figura 9.26. Il processo di calibrazione viene spesso condotto parallelamente al processo di messa a punto per QAT.

Figura 9.26: Quantization-Aware Training. Fonte: Gholami et al. (2021).

La “Quantization-Aware Training” funge da estensione naturale della “Post-Training Quantization”. Dopo la quantizzazione iniziale eseguita da PTQ, QAT viene utilizzata per perfezionare e mettere a punto ulteriormente i parametri quantizzati: vedere come in Figura 9.27, il modello PTQ subisce un ulteriore passaggio, QAT. Comporta un processo di riqualificazione in cui il modello viene esposto a ulteriori iterazioni di training utilizzando i dati originali. Questo approccio di training dinamico consente al modello di adattare e regolare i suoi parametri, compensando il degrado delle prestazioni causato dalla quantizzazione.

Figura 9.27: PTQ e QAT. Fonte: «The Ultimate Guide to Deep Learning Model Quantization and Quantization-Aware Training» (s.d.).

Figura 9.28 mostra l’accuratezza relativa di diversi modelli dopo PTQ e QAT. In quasi tutti i casi, QAT produce un’accuratezza migliore di PTQ. Si consideri ad esempio EfficientNet b0. Dopo PTQ, l’accuratezza scende dal 76.85% a 72.06%. Ma quando applichiamo QAT, l’accuratezza rimbalza al 76.95% (con persino un leggero miglioramento rispetto all’accuratezza originale).

Model	fp32 Accuracy	PTQ best Calibration	PTQ best Accuracy	PTQ best Relative	QAT Accuracy	QAT Relative
MobileNet v1	71.88	99.9%	70.39	-2.07%	72.07	0.26%
MobileNet v2	71.88	99.99%	71.14	-1.03%	71.56	-0.45%
ResNet50 v1.5	76.16	Entropy	76.05	-0.14%	76.85	0.91%
ResNet152 v1.5	78.32	Entropy	78.21	-0.14%	78.61	0.37%
Inception v3	77.34	Entropy	77.54	0.26%	78.43	1.41%
Inception v4	79.71	99.99%	79.63	-0.10%	80.14	0.54%
ResNeXt50	77.61	Entropy	77.46	-0.19%	77.67	0.08%
ResNeXt101	79.30	99.999%	79.17	-0.16%	79.01	-0.37%
EfficientNet b0	76.85	Entropy	72.06	-6.23%	76.95	0.13%
EfficientNet b3	81.61	99.99%	80.28	-1.63%	81.07	-0.66%
Faster R-CNN	36.95	Entropy	36.82	-0.35%	36.76	-0.51%
Mask R-CNN	37.89	99.9999%	37.80	-0.24%	37.75	-0.37%
Retinanet	39.30	99.999%	39.19	-0.28%	39.25	-0.13%
FCN	63.70	Entropy	64.00	0.47%	64.10	0.63%
DeepLabV3	67.40	99.999%	67.50	0.15%	67.50	0.15%
GNMT	24.27	Entropy	24.53	1.07%	24.38	0.45%
Transformer	28.27	99.99%	27.71	-1.98%	28.21	-0.21%
Jasper	96.09	Entropy	96.11	0.02%	96.10	0.01%
BERT Large	91.01	99.999%	90.20	-0.89%	90.67	-0.37%

Figura 9.28: Accuratezza relativa di PTQ e QAT. Fonte: Wu, Judd, e Isaev (2020).

9.3.8 Pesi vs. Attivazioni

Quantizzazione del peso: Comporta la conversione dei pesi continui o ad alta precisione di un modello in pesi a bassa precisione, come la conversione dei pesi Float32 in pesi INT8 (interi) quantizzati - in Figura 9.29, la quantizzazione del peso avviene nel secondo passaggio (quadrati rossi) quando moltiplichiamo gli input. Ciò riduce le dimensioni del modello, riducendo così la memoria richiesta per archiviare il modello e le risorse computazionali necessarie per eseguire l'inferenza. Ad esempio, si consideri una matrice di pesi in un layer di rete neurale con pesi Float32 come [0.215, -1.432, 0.902, ...]. Attraverso la quantizzazione del peso, questi potrebbero essere mappati su valori INT8 come [27, -183, 115, ...], riducendo significativamente la memoria richiesta per memorizzarli.

Figura 9.29: Quantizzazione del peso e dell'attivazione. Fonte: HarvardX.

Quantizzazione dell'Attivazione: Comporta la quantizzazione dei valori di attivazione (output dei livelli) durante l'inferenza del modello. Ciò può ridurre le risorse computazionali

richieste durante l'inferenza, ma introduce ulteriori problemi nel mantenimento dell'accuratezza del modello a causa della ridotta precisione dei calcoli intermedi. Ad esempio, in una rete neurale convoluzionale (CNN), le mappe di attivazione (mappe delle feature) prodotte dai layer convoluzionali, originariamente in Float32, potrebbero essere quantizzate su INT8 durante l'inferenza per accelerare il calcolo, in particolare su hardware ottimizzato per l'aritmetica degli interi. Inoltre, un lavoro recente ha esplorato l'uso della quantizzazione del "Activation-aware Weight Quantization" per la compressione e l'accelerazione LLM, che comporta la protezione di solo l'1% dei pesi salienti più importanti osservando le attivazioni, non i pesi (Lin et al. 2023).

9.3.9 Compromessi

La quantizzazione introduce invariabilmente un compromesso tra dimensioni/prestazioni del modello e accuratezza. Sebbene riduca significativamente l'ingombro della memoria e possa accelerare l'inferenza, specialmente su hardware ottimizzato per aritmetica a bassa precisione, la precisione ridotta può degradare l'accuratezza del modello.

Dimensioni del Modello: Un modello con pesi rappresentati come Float32 quantizzato a INT8 può teoricamente ridurre le dimensioni del modello di un fattore 4, consentendone l'implementazione su dispositivi con memoria limitata. Le dimensioni di grandi modelli linguistici si stanno sviluppando a un ritmo più veloce della memoria GPU negli ultimi anni, portando a un grande divario tra domanda e offerta di memoria. Figura 9.30 illustra la recente tendenza del divario crescente tra le dimensioni del modello (linea rossa) e la memoria dell'acceleratore (linea gialla). Le tecniche di quantizzazione e compressione del modello possono aiutare a colmare il divario

Figura 9.30: Dimensioni del modello vs. memoria dell'acceleratore. Fonte: Xiao et al. (2022).

Velocità di Inferenza: La quantizzazione può anche accelerare l'inferenza, poiché l'aritmetica a precisione inferiore è computazionalmente meno costosa. Ad esempio, alcuni acceleratori hardware, come Edge TPU di Google, sono ottimizzati per l'aritmetica INT8 e possono eseguire l'inferenza in modo significativamente più rapido con modelli quantizzati INT8 rispetto alle loro controparti in virgola mobile. La riduzione della memoria dalla quantizzazione aiuta a ridurre la quantità di trasmissione dei dati, risparmiando memoria e

velocizzando il processo. Figura 9.31 confronta l'aumento della produttività e la riduzione della memoria della larghezza di banda per diversi tipi di dati sulla NVIDIA Turing GPU.

Input Data type	Accumulation Data type	Math Throughput	Bandwidth Reduction
FP32	FP32	1x	1x
FP16	FP16	8x	2x
INT8	INT32	16x	4x
INT4	INT32	32x	8x
INT1	INT32	128x	32x

Figura 9.31: Vantaggi dei tipi di dati a precisione inferiore. Fonte: Wu, Judd, e Isaev (2020).

Precisione: La riduzione della precisione numerica post-quantizzazione può portare a un degrado della precisione del modello, che potrebbe essere accettabile in alcune applicazioni (ad esempio, classificazione delle immagini) ma non in altre (ad esempio, diagnosi medica). Pertanto, dopo la quantizzazione, il modello richiede in genere una ricalibrazione o una messa a punto per mitigare la perdita di accuratezza. Inoltre, un lavoro recente ha esplorato l'uso di Activation-aware Weight Quantization (Lin et al. 2023) che si basa sull'osservazione che proteggere solo l'1% dei pesi salienti può ridurre notevolmente l'errore di quantizzazione.

9.3.10 Quantizzazione e Potatura

Pruning [potatura] e quantizzazione funzionano bene insieme ed è stato scoperto che il pruning non ostacola la quantizzazione. In effetti, il pruning può aiutare a ridurre l'errore di quantizzazione. Intuitivamente, ciò è dovuto al pruning che riduce il numero di pesi da quantizzare, riducendo così l'errore accumulato dalla quantizzazione. Ad esempio, una AlexNet non potata ha 60 milioni di pesi da quantizzare mentre una AlexNet potata ha solo 6.7 milioni di pesi da quantizzare. Questa significativa riduzione dei pesi aiuta a ridurre l'errore tra la quantizzazione dell'AlexNet non potato rispetto all'AlexNet potato. Inoltre, studi recenti hanno scoperto che il pruning consapevole della quantizzazione genera modelli più efficienti dal punto di vista computazionale rispetto al pruning o alla quantizzazione da soli; in genere, ha prestazioni simili o migliori in termini di efficienza computazionale rispetto ad altre tecniche di ricerca dell'architettura neurale come l'ottimizzazione bayesiana (Hawks et al. 2021).

Figura 9.32: Precisione rispetto al tasso di compressione con diversi metodi di compressione. Fonte: Han, Mao, e Dally (2015).

9.3.11 Quantizzazione Edge-aware

La quantizzazione non solo riduce le dimensioni del modello, ma consente anche calcoli più rapidi e consuma meno energia, rendendola fondamentale per lo sviluppo per edge. I dispositivi edge in genere hanno vincoli di risorse rigidi con elaborazione, memoria e potenza, impossibili da soddisfare per molti dei modelli deep NN profondi odierni. Inoltre, i processori edge non supportano le operazioni in virgola mobile, rendendo la quantizzazione intera particolarmente importante per chip come GAP-8, un SoC RISC-V per l'inferenza edge con un acceleratore CNN dedicato, che supporta solo l'aritmetica intera.
Una piattaforma hardware che utilizza la quantizzazione è il gruppo ARM Cortex-M di core di processori ARM RISC a 32 bit. Sfruttano la quantizzazione a virgola fissa con fattori di scala di potenza di due, in modo che la quantizzazione e la de-quantizzazione possano essere eseguite in modo efficiente tramite spostamento di bit. Inoltre, Google Edge TPU, la soluzione emergente di Google per l'esecuzione di inferenze in periferia, è progettata per dispositivi piccoli e a bassa potenza e può supportare solo l'aritmetica a 8 bit. Molti modelli di reti neurali complesse che potevano essere distribuiti solo su server a causa delle loro elevate esigenze di elaborazione possono ora essere eseguiti su dispositivi edge grazie ai recenti progressi (ad esempio metodi di quantizzazione) nel campo dell'edge computing.
Oltre a essere una tecnica indispensabile per molti processori edge, la quantizzazione ha anche apportato notevoli miglioramenti ai processori non edge, incoraggiando tali processori a soddisfare i requisiti del Service Level Agreement (SLA) come la latenza del 99° percentile. Pertanto, la quantizzazione combinata con una logica efficiente a bassa precisione e acceleratori dedicati di deep learning, è stata una forza trainante cruciale per l'evoluzione di tali processori edge.
Video 9.1 è una lezione sulla quantizzazione e sui diversi metodi di quantizzazione.

Video 9.1: Quantizzazione

https://www.youtube.com/watch?v=AlASZb93rrc

9.4 Implementazione Hardware Efficiente

L'implementazione hardware efficiente trascende la selezione di componenti adatti; richiede una comprensione olistica di come il software interagirà con le architetture sottostanti. L'essenza del raggiungimento delle massime prestazioni nelle applicazioni TinyML non risiede solo nell'affinare gli algoritmi per l'hardware, ma anche nell'assicurare che l'hardware sia strategicamente adattato per supportare questi algoritmi. Questa sinergia tra hardware e software è fondamentale. Mentre esaminiamo più a fondo le complessità dell'implementazione hardware efficiente, il significato di un approccio di progettazione congiunta, in cui hardware e software vengono sviluppati in tandem, diventa sempre più evidente. Questa sezione fornisce una panoramica delle tecniche di come l'hardware e le interazioni tra hardware e software possono essere ottimizzati per migliorare le prestazioni dei modelli.

9.4.1 Ricerca di Architettura Neurale Basata sull'Hardware

Concentrarsi solo sulla precisione durante l'esecuzione della ricerca di architettura neurale porta a modelli esponenzialmente complessi e che richiedono memoria e capacità di elaborazione crescenti. Ciò ha portato a vincoli hardware che limitano lo sfruttamento dei modelli di apprendimento profondo al loro pieno potenziale. Progettare manualmente l'architettura del modello è ancora più difficile se si considerano la varietà e le limitazioni dell'hardware. Ciò ha portato alla creazione di Hardware-aware Neural Architecture Search che incorpora le contrazioni hardware nella loro ricerca e ottimizza lo spazio di ricerca per un hardware e una precisione specifici. HW-NAS può essere categorizzato in base a come

ottimizza per l'hardware. Esploreremo brevemente queste categorie e lasceremo dei link a documenti correlati per il lettore interessato.

9.4.1.1 Configurazione Single Target, Fixed Platfrom

L'obiettivo qui è trovare la migliore architettura in termini di precisione ed efficienza hardware per un hardware target fisso. Per un hardware specifico, ad esempio Arduino Nicla Vision, questa categoria di HW-NAS cercherà l'architettura che ottimizza precisione, latenza, consumo energetico, ecc.

9.4.1.1.1 Strategia di Ricerca Hardware-aware

Qui, la ricerca è un problema di ottimizzazione multi-obiettivo, in cui sia l'accuratezza che il costo dell'hardware guidano l'algoritmo di ricerca per trovare l'architettura più efficiente (Tan et al. 2019; H. Cai, Zhu, e Han 2019; B. Wu et al. 2019).

9.4.1.1.2 Spazio di Ricerca Hardware-aware

Qui, lo spazio di ricerca è limitato alle architetture che funzionano bene sull'hardware specifico. Questo può essere ottenuto misurando le prestazioni degli operatori (operatore Conv, operatore Pool, ...) o definendo un set di regole che limitano lo spazio di ricerca. (L. L. Zhang et al. 2020)

9.4.1.2 Configurazioni Single Target, Multiple Platform

Alcuni hardware possono avere configurazioni diverse. Ad esempio, gli FPGA hanno blocchi logici configurabili (CLB) che possono essere configurati dal firmware. Questo metodo consente all'HW-NAS di esplorare diverse configurazioni. (Y. Hu et al. 2023; Ho Yoon et al. 2012)

9.4.1.3 Target Multipli

Questa categoria mira a ottimizzare un singolo modello per più hardware. Questo può essere utile per lo sviluppo di dispositivi mobili in quanto può ottimizzare diversi modelli di telefoni. (Chu et al. 2021; Y. Hu et al. 2023)

9.4.1.4 Esempi di "Hardware-Aware Neural Architecture Search"

9.4.1.4.1 TinyNAS

TinyNAS adotta un approccio in due fasi per trovare un'architettura ottimale per il modello tenendo a mente i vincoli del microcontrollore specifico.
Innanzitutto, TinyNAS genera più spazi di ricerca variando la risoluzione di input del modello e il numero di canali dei layer. Quindi, TinyNAS sceglie uno spazio di ricerca in base ai FLOP (operazioni in virgola mobile al secondo) di ogni spazio di ricerca. Gli spazi con una probabilità maggiore di contenere architetture con un numero elevato di FLOP producono modelli con maggiore accuratezza: confrontare la linea rossa con la linea nera in Figura 9.33. Poiché un numero maggiore di FLOP significa che il modello ha una maggiore capacità di calcolo, è più probabile che il modello abbia una maggiore accuratezza.
Poi, TinyNAS esegue un'operazione di ricerca sullo spazio scelto per trovare l'architettura ottimale per i vincoli specifici del microcontrollore. (J. Lin et al. 2020)

Figura 9.33: Precisione degli spazi di ricerca. Fonte: J. Lin et al. (2020).

9.4.1.5 Topology-Aware NAS

Si concentra sulla creazione e l'ottimizzazione di uno spazio di ricerca allineato alla topologia hardware del dispositivo. (T. Zhang et al. 2020)

9.4.2 Sfide nella "Hardware-Aware Neural Architecture Search"

Sebbene HW-NAS abbia un potenziale elevato per trovare architetture ottimali per TinyML, presenta alcuni problemi. Le metriche hardware come latenza, consumo energetico e utilizzo dell'hardware sono più difficili da valutare rispetto alle metriche di accuratezza o di perdita. Spesso richiedono strumenti specializzati per misure precise. Inoltre, l'aggiunta di tutte queste metriche porta a uno spazio di ricerca molto più grande. Ciò fa sì che HW-NAS sia dispendioso in termini di tempo e denaro. Deve essere applicato a ogni hardware per risultati ottimali, tra le altre cose, il che significa che se si deve distribuire il modello su più dispositivi, la ricerca deve essere condotta più volte e produrrà modelli diversi, a meno che non si ottimizzi per tutti, il che significa una minore accuratezza. Infine, l'hardware cambia frequentemente e potrebbe essere necessario eseguire HW-NAS su ogni versione.

9.4.3 Ottimizzazioni del Kernel

Le ottimizzazioni del kernel sono modifiche apportate al kernel per migliorare le prestazioni dei modelli di apprendimento automatico su dispositivi con risorse limitate. Separeremo le ottimizzazioni del kernel in due tipi.

9.4.3.1 Ottimizzazioni del kernel Generali

Queste sono ottimizzazioni del kernel da cui tutti i dispositivi possono trarre vantaggio. Forniscono tecniche per convertire il codice in istruzioni più efficienti.

9.4.3.1.1 "Srotolamento" del Loop

Invece di avere un loop con "loop control" (incrementando il contatore, si controlla la condizione di terminazione del loop), il loop può essere srotolato e il sovraccarico del "loop control" può essere omesso. Questo può anche fornire ulteriori opportunità di parallelismo che potrebbero non essere possibili con la struttura con loop. Questo può essere particolarmente utile per loop stretti, in cui il corpo del loop è un piccolo numero di istruzioni con molte iterazioni.

9.4.3.1.2 Blocking

Il Blocking viene utilizzato per rendere più efficienti i pattern di accesso alla memoria. Se abbiamo tre calcoli, il primo e l'ultimo devono accedere alla cache A e il secondo deve accedere alla cache B, il "blocking" ferma i primi due calcoli per ridurre il numero di letture di memoria necessarie.

9.4.3.1.3 Tiling

Analogamente al blocking, il tiling [piastrellatura] divide i dati e il calcolo in blocchi, ma si estende oltre i miglioramenti della cache. Il tiling crea partizioni di calcolo indipendenti che possono essere eseguite in parallelo, il che può comportare significativi miglioramenti delle prestazioni.

9.4.3.1.4 Librerie Kernel Ottimizzate

Questo comprende lo sviluppo di kernel ottimizzati che sfruttano appieno un hardware specifico. Un esempio è la libreria CMSIS-NN, che è una raccolta di kernel di reti neurali efficienti sviluppati per ottimizzare le prestazioni e ridurre al minimo l'ingombro di memoria dei modelli sui processori Arm Cortex-M, comuni sui dispositivi edge IoT. Il kernel sfrutta più capacità hardware dei processori Cortex-M come Single Instruction Multiple Data (SIMD), Floating Point Unit (FPU) e M-Profile Vector Extensions (MVE). Queste ottimizzazioni rendono più efficienti le operazioni comuni come le moltiplicazioni di matrici, aumentando le prestazioni delle operazioni del modello sui processori Cortex-M. (Lai, Suda, e Chandra 2018b)

9.4.4 Compute-in-Memory (CiM)

Questo è un esempio di progettazione congiunta di algoritmo e hardware. CiM è un paradigma di elaborazione che esegue calcoli all'interno della memoria. Pertanto, le architetture CiM consentono di eseguire operazioni direttamente sui dati archiviati, senza la necessità di spostare i dati avanti e indietro tra unità di elaborazione e memoria separate. Questo paradigma di progettazione è particolarmente utile in scenari in cui lo spostamento dei dati è una fonte primaria di consumo energetico e latenza, come nelle applicazioni TinyML su dispositivi edge. Figura 9.34 è un esempio di utilizzo di CiM in TinyML: l'individuazione delle parole chiave richiede un processo sempre attivo che cerca determinate parole di attivazione (come "Hey, Siri"). Data la natura ad alta intensità di risorse di questa attività, l'integrazione di CiM per il modello di rilevamento delle parole chiave sempre attivo può migliorare l'efficienza.

Attraverso la progettazione congiunta di algoritmo e hardware, gli algoritmi possono essere ottimizzati per sfruttare le caratteristiche uniche delle architetture CiM e, l'hardware CiM può essere personalizzato o configurato per supportare meglio i requisiti di elaborazione e le caratteristiche degli algoritmi. Ciò si ottiene utilizzando le proprietà analogiche delle celle di memoria, come l'addizione e la moltiplicazione nella DRAM. (C. Zhou et al. 2021)

Figura 9.34: CiM per l'individuazione delle parole chiave. Fonte: C. Zhou et al. (2021).

9.4.5 Ottimizzazione dell'Accesso alla Memoria

Dispositivi diversi possono avere gerarchie di memorie diverse. L'ottimizzazione per la gerarchia di memoria specifica nell'hardware specifico può portare a grandi miglioramenti delle prestazioni riducendo le costose operazioni di lettura e scrittura nella memoria.

L'ottimizzazione del flusso di dati può essere ottenuta ottimizzando il riutilizzo dei dati all'interno di un singolo layer e tra più layer. Questa ottimizzazione del flusso di dati può essere adattata alla gerarchia di memoria specifica dell'hardware, il che può portare a maggiori vantaggi rispetto alle ottimizzazioni generali per diversi hardware.

9.4.5.1 Sfruttamento dei Dati Sparsi

Il Pruning [potatura] è un approccio fondamentale per comprimere i modelli e renderli compatibili con dispositivi con risorse limitate. Ciò si traduce in modelli sparsi in cui molti pesi sono 0. Pertanto, sfruttare questa diradazione può portare a miglioramenti significativi nelle prestazioni. Sono stati creati degli strumenti per ottenere esattamente questo. RAMAN, è un acceleratore TinyML sparse progettato per l'inferenza su dispositivi edge. RAMAN sovrappone le attivazioni di input e output sullo stesso spazio di memoria, riducendo i requisiti di archiviazione fino al 50%. (Krishna et al. 2023)

9.4.5.2 Framework di Ottimizzazione

I framework di ottimizzazione sono stati introdotti per sfruttare le capacità specifiche dell'hardware per accelerare il software. Un esempio di tale framework è hls4ml: Figura 9.35 fornisce una panoramica del flusso di lavoro del framework. Questo flusso di lavoro di co-progettazione software-hardware open source aiuta a interpretare e tradurre algoritmi di machine learning per l'implementazione con tecnologie FPGA e ASIC. Funzionalità quali ottimizzazione di rete, nuove API Python, potatura consapevole della quantizzazione e flussi di lavoro FPGA end-to-end sono integrate nel framework hls4ml, sfruttando unità di elaborazione parallele, gerarchie di memoria e set di istruzioni specializzati per ottimizzare i modelli per hardware edge. Inoltre, hls4ml è in grado di tradurre algoritmi di apprendimento automatico direttamente nel firmware FPGA.

Figura 9.35: workflow del framework hls4ml. Fonte: Fahim et al. (2021).

Un altro framework per FPGA che si concentra su un approccio olistico è CFU Playground (Prakash, Callahan, et al. 2023)

9.4.5.3 Hardware Costruito Attorno al Software

In un approccio contrastante, l'hardware può essere progettato su misura attorno ai requisiti software per ottimizzare le prestazioni per un'applicazione specifica. Questo paradigma crea hardware specializzato per adattarsi meglio alle specifiche del software, riducendo così il sovraccarico computazionale e migliorando l'efficienza operativa. Un esempio di questo approccio è un'applicazione di riconoscimento vocale di (J. Kwon e Park 2021). Il documento propone una struttura in cui le operazioni di pre-elaborazione, tradizionalmente gestite dal

software, sono assegnate ad un hardware progettato su misura. Questa tecnica è stata ottenuta introducendo la logica resistore-transistor in un modulo audio a circuito inter-integrato per il windowing e l'acquisizione di dati audio grezzi nell'applicazione di riconoscimento vocale. Di conseguenza, questa "delega" delle operazioni di pre-elaborazione ha portato a una riduzione del carico computazionale sul software, mostrando un'applicazione pratica della creazione di hardware attorno al software per migliorare l'efficienza e le prestazioni.

Figura 9.36: Delega dell'elaborazione dei dati a un FPGA. Fonte: J. Kwon e Park (2021).

9.4.5.4 SplitNet

Li SplitNet sono state introdotte nel contesto dei sistemi Head-Mounted. Distribuiscono il carico di lavoro delle Deep Neural Network (DNN) tra i sensori della telecamera e un aggregatore. Ciò è particolarmente interessante nel contesto di TinyML. Il framework SplitNet è un NAS split-aware per trovare l'architettura di rete neurale ottimale per ottenere una buona accuratezza, dividere il modello tra i sensori e l'aggregatore e ridurre al minimo la comunicazione tra i sensori e l'aggregatore.

Figura 9.37 dimostra come le SplitNet (in rosso) ottengano una maggiore accuratezza per una latenza inferiore (in esecuzione su ImageNet) rispetto ad altri approcci, come l'esecuzione del DNN sul sensore (All-on-sensor; in verde) o sul cellulare (All-on-aggregator; in blu). La comunicazione minima è importante in TinyML dove la memoria è fortemente limitata, in questo modo i sensori conducono parte dell'elaborazione sui loro chip e poi inviano solo le informazioni necessarie all'aggregatore. Durante i test su ImageNet, SplitNets è stato in grado di ridurre la latenza di un ordine di grandezza sui dispositivi di visione artificiale montati sulla testa [occhiali o visori]. Ciò può essere utile quando il sensore ha il suo chip. (Dong et al. 2022)

Figura 9.37: Le SplitNet rispetto ad altri approcci. Fonte: Dong et al. (2022).

9.4.5.5 Hardware Specifico per il "Data Augmentation"

Ogni dispositivo edge può possedere caratteristiche di sensore uniche, che portano a specifici pattern di rumore che possono influire sulle prestazioni del modello. Un esempio sono i dati audio, in cui sono prevalenti le variazioni derivanti dalla scelta del microfono. Applicazioni come le Keyword Spotting possono sperimentare miglioramenti sostanziali incorporando dati registrati da dispositivi simili a quelli destinati all'implementazione. La messa a punto dei modelli esistenti può essere impiegata per adattare i dati in modo preciso alle caratteristiche distintive del sensore.

9.5 Supporto Software e Framework

Sebbene tutte le tecniche sopra menzionate come pruning, quantizzazione e numeri efficienti siano ben note, rimarrebbero poco pratiche e inaccessibili senza un ampio supporto software. Ad esempio, la quantizzazione diretta di pesi e attivazioni in un modello richiederebbe la modifica manuale della definizione del modello e l'inserimento di operazioni di quantizzazione. Allo stesso modo, la potatura diretta dei pesi del modello richiede la manipolazione dei tensori dei pesi. Tali approcci noiosi diventano impraticabili su larga scala. Senza l'ampia innovazione software nei framework, negli strumenti di ottimizzazione e nell'integrazione hardware, la maggior parte di queste tecniche rimarrebbe teorica o praticabile solo per gli esperti. Senza API del framework e automazione per semplificare l'applicazione di queste ottimizzazioni, non verrebbero adottate. Il supporto software le rende accessibili al pubblico e sblocca vantaggi concreti. Inoltre, problemi come la messa a punto degli iperparametri per la potatura, la gestione del compromesso tra dimensioni del modello e accuratezza e la garanzia della compatibilità con i dispositivi target pongono ostacoli che gli sviluppatori devono superare.

9.5.1 API Native di Ottimizzazione

I principali framework di machine learning come TensorFlow, PyTorch e MXNet forniscono librerie e API per consentire l'applicazione di tecniche comuni di ottimizzazione dei modelli senza richiedere implementazioni personalizzate. Ad esempio, TensorFlow offre il TensorFlow Model Optimization Toolkit che contiene moduli come:

- **Quantization**: Applica un training che tiene conto della quantizzazione per convertire i modelli in virgola mobile in una precisione inferiore come int8 con una perdita di accuratezza minima. Gestisce la quantizzazione del peso e dell'attivazione.
- **Sparsity**: Fornisce API di potatura per indurre la "sparsità" e rimuovere connessioni non necessarie in modelli come le reti neurali. Può potare pesi, livelli, ecc.
- **Clustering**: Supporta la compressione del modello raggruppando i pesi per tassi di compressione più elevati.

Queste API consentono agli utenti di abilitare tecniche di ottimizzazione come la quantizzazione e la potatura senza modificare direttamente il codice del modello. È possibile configurare parametri come i tassi di "sparsità" del target, le larghezze di bit di quantizzazione, ecc. Allo stesso modo, PyTorch fornisce torch.quantization per convertire i modelli in rappresentazioni di precisione inferiore. TorchTensor e TorchModule formano le classi di base per il supporto della quantizzazione. Offre inoltre torch.nn.utils.prune per la potatura nativa dei modelli. MXNet offre layer gluon.contrib che aggiungono funzionalità di quantizzazione come l'arrotondamento a punto fisso e l'arrotondamento stocastico di pesi/attivazioni durante l'addestramento. Ciò consente di includere facilmente la quantizzazione nei modelli gluon.

Il vantaggio principale delle ottimizzazioni integrate è che gli utenti possono applicarle senza dover reimplementare tecniche complesse. Ciò rende i modelli ottimizzati accessibili a un'ampia gamma di professionisti. Garantisce inoltre che le best practice siano seguite

basandosi sulla ricerca e sull'esperienza nell'implementazione dei metodi. Man mano che emergono nuove ottimizzazioni, i framework si sforzano di fornire supporto nativo e API ove possibile per abbassare ulteriormente la barriera verso un ML efficiente. La disponibilità di questi strumenti è fondamentale per un'adozione diffusa.

9.5.2 Strumenti di Ottimizzazione Automatizzata

Gli strumenti di ottimizzazione automatizzati forniti dai framework possono analizzare i modelli e applicare automaticamente ottimizzazioni come quantizzazione, potatura e fusione degli operatori per rendere il processo più semplice e accessibile senza un'eccessiva messa a punto manuale. In effetti, questo si basa sulla sezione precedente. Ad esempio, TensorFlow fornisce il TensorFlow Model Optimization Toolkit che contiene moduli come:

- **QuantizationAwareTraining**: Quantizza automaticamente pesi e attivazioni in un modello per ridurre la precisione come UINT8 o INT8 con una perdita di accuratezza minima. Inserisce nodi di quantizzazione falsi durante l'addestramento in modo che il modello possa imparare a essere compatibile con la quantizzazione.
- **Pruning**: Rimuove automaticamente le connessioni non necessarie in un modello in base all'analisi dell'importanza del peso. Può potare interi filtri in livelli convoluzionali o "attention head" [teste di attenzione] nei trasformatori. Gestisce il ri-addestramento iterativo per recuperare qualsiasi perdita di accuratezza.
- **GraphOptimizer**: Applica ottimizzazioni grafiche come la fusione degli operatori per consolidare le operazioni e ridurre la latenza di esecuzione, in particolare per l'inferenza. In Figura 9.38, si può vedere l'originale (Source Graph) a sinistra e come le sue operazioni vengono trasformate (consolidate) a destra. Notare come Block1 in Source Graph abbia 3 passaggi separati (Convolution, BiasAdd e Activation), che vengono poi consolidati insieme in Block1 su Optimized Graph.

Figura 9.38: GraphOptimizer. Fonte: Wess et al. (2020).

Questi moduli automatizzati richiedono solo all'utente di fornire il modello originale in virgola mobile e di gestire la pipeline di ottimizzazione end-to-end, inclusa qualsiasi riqualificazione per ripristinare la precisione. Anche altri framework come PyTorch offrono un crescente supporto all'automazione, ad esempio tramite torch.quantization.quantize_dynamic. L'ottimizzazione automatizzata rende l'apprendimento automatico efficiente accessibile ai professionisti senza competenze di ottimizzazione.

9.5.3 Librerie di Ottimizzazione Hardware

Librerie hardware come TensorRT e TensorFlow XLA consentono di ottimizzare i modelli per l'hardware target tramite tecniche di cui abbiamo discusso in precedenza.

- **Quantizzazione:** Ad esempio, TensorRT e TensorFlow Lite supportano entrambi la quantizzazione dei modelli durante la conversione nel loro formato. Ciò fornisce accelerazioni sui SoC mobili con supporto INT8/INT4.
- **Ottimizzazione del Kernel:** ad esempio, TensorRT esegue l'auto-tuning per ottimizzare i kernel CUDA in base all'architettura GPU per ogni layer nel grafo del modello. Ciò estrae la massima produttività.
- **Fusione degli Operatori:** TensorFlow XLA esegue una fusione aggressiva per creare un binario ottimizzato per le TPU. Sui dispositivi mobili, framework come NCNN supportano anche operatori fusi [unificati].
- **Codice Specifico per l'Hardware:** Le librerie vengono utilizzate per generare codice binario ottimizzato specializzato per l'hardware target. Per esempio, TensorRT usa librerie Nvidia CUDA/cuDNN che sono ottimizzate manualmente per ogni architettura GPU. Questa codifica specifica per hardware è fondamentale per le prestazioni. Sui dispositivi TinyML, questo può significare codice assembly ottimizzato per una CPU Cortex M4, ad esempio. I fornitori forniscono CMSIS-NN e altre librerie.
- **Ottimizzazioni del Layout dei Dati:** Possiamo sfruttare in modo efficiente la gerarchia della memoria di hardware come cache e registri tramite tecniche come riorganizzazione tensore/peso, tiling e riutilizzo. Ad esempio, TensorFlow XLA ottimizza i layout dei buffer per massimizzare l'utilizzo della TPU. Questo aiuta qualsiasi sistema con limiti di memoria.
- **Ottimizzazione Basata sulla Profilazione:** Possiamo usare strumenti di profilazione per identificare i colli di bottiglia. Ad esempio, regolare i livelli di fusione del kernel in base alla profilazione della latenza. Sui SoC mobili, fornitori come Qualcomm forniscono profiler in SNPE per trovare opportunità di ottimizzazione nelle CNN. Questo approccio basato sui dati è importante per le prestazioni.

Integrando i modelli di framework con queste librerie hardware tramite pipeline di conversione ed esecuzione, gli sviluppatori di ML possono ottenere significativi incrementi di velocità e guadagni di efficienza da ottimizzazioni di basso livello su misura per l'hardware target. La stretta integrazione tra software e hardware è fondamentale per consentire un'implementazione performante delle applicazioni di ML, in particolare su dispositivi mobili e TinyML.

9.5.4 Visualizzazione delle Ottimizzazioni

L'implementazione di tecniche di ottimizzazione del modello senza visibilità degli effetti sul modello può essere impegnativa. Strumenti dedicati o strumenti di visualizzazione possono fornire informazioni critiche e utili sulle modifiche del modello e aiutano a tracciare il processo di ottimizzazione. Consideriamo le ottimizzazioni che abbiamo considerato in precedenza, come la potatura per la "sparsity" [diradazione] e la quantizzazione.

9.5.4.0.1 Sparsità

Ad esempio, si considerino le ottimizzazioni di sparsity. Gli strumenti di visualizzazione di sparsity possono fornire informazioni critiche sui modelli potati, mappando esattamente quali pesi sono stati rimossi. Ad esempio, le mappe di calore di sparsity possono utilizzare gradienti di colore per indicare la percentuale di pesi potati in ogni layer di una rete neurale. I layer con percentuali di potatura più elevate appaiono più scuri (cfr. Figura 9.39). Questo identifica quali layer sono stati semplificati di più tramite potatura (Souza 2020).

Figura 9.39: Mappa "termica" della rete sparsa. Fonte: Numenta.

I grafici di tendenza possono anche tracciare la scarsità nei successivi round di potatura: possono mostrare una rapida potatura iniziale seguita da incrementi più graduali. Il tracciamento della diradazione globale corrente insieme a statistiche come la diradazione media, minima e massima per ogli layer in tabelle o grafici fornisce una panoramica della composizione del modello. Per una rete convoluzionale di esempio, questi strumenti potrebbero rivelare che il primo layer di convoluzione viene potato del 20% mentre quello di classificazione finale viene potato del 70% data la sua ridondanza. La diradazione del modello globale può aumentare dal 10% dopo la potatura iniziale al 40% dopo cinque round. Rendendo i dati di diradazione visivamente accessibili, i professionisti possono comprendere meglio esattamente come il loro modello viene ottimizzato e quali aree vengono interessate. La visibilità consente loro di mettere a punto e controllare il processo di potatura per una determinata architettura.
La visualizzazione della diradazione trasforma la potatura in una tecnica trasparente anziché in un'operazione "black-box".

9.5.4.0.2 Quantizzazione

La conversione di modelli in precisioni numeriche inferiori tramite quantizzazione introduce errori che possono influire sulla precisione del modello se non vengono monitorati e affrontati correttamente. La visualizzazione delle distribuzioni degli errori di quantizzazione fornisce informazioni preziose sugli effetti dei numeri di precisione ridotti applicati a diverse parti di un modello. Per questo, è possibile generare istogrammi degli errori di quantizzazione per pesi e attivazioni. Questi istogrammi possono rivelare la forma della distribuzione degli errori, se assomigliano a una distribuzione gaussiana o contengono valori anomali e picchi significativi. Figura 9.40 mostra le distribuzioni di diversi metodi di quantizzazione. Valori anomali elevati possono indicare problemi con particolari layer che gestiscono la quantizzazione. Il confronto degli istogrammi tra layer evidenzia eventuali aree problematiche che si distinguono con errori anormalmente elevati.

Figura 9.40: Errori di Quantizzazione. Fonte: Kuzmin et al. (2022).

Le visualizzazioni di attivazione sono importanti anche per rilevare problemi di overflow. Con la mappatura a colori delle attivazioni prima e dopo la quantizzazione, tutti i valori spinti al di fuori degli intervalli previsti diventano visibili. Ciò rivela problemi di saturazione e troncamento che potrebbero alterare le informazioni che fluiscono attraverso il modello. Il rilevamento di questi errori consente di ricalibrare le attivazioni per evitare la perdita di informazioni (Mandal 2022). Figura 9.41 è una mappatura a colori dei kernel convoluzionali AlexNet.

Figura 9.41: Mappatura a colori delle attivazioni. Fonte: Krizhevsky, Sutskever, e Hinton (2017).

Altre tecniche, come il tracciamento dell'errore di quantizzazione quadratico medio complessivo a ogni passaggio del processo di addestramento consapevole della quantizzazione, identificano fluttuazioni e divergenze. Picchi improvvisi nel grafico di tracciamento possono indicare punti in cui la quantizzazione sta interrompendo l'addestramento del modello. Il monitoraggio di questa metrica crea intuizione sul comportamento del modello in fase di quantizzazione. Insieme, queste tecniche trasformano la quantizzazione in un processo trasparente. Le intuizioni empiriche consentono ai professionisti di valutare correttamente gli effetti della quantizzazione. Individuano le aree dell'architettura del modello o del processo di training da ricalibrare in base ai problemi di quantizzazione osservati. Ciò aiuta a ottenere modelli quantizzati numericamente stabili e accurati.

Fornire questi dati consente ai professionisti di valutare correttamente l'impatto della quantizzazione e identificare potenziali aree problematiche del modello da ricalibrare o riprogettare per renderlo più adatto alla quantizzazione. Questa analisi empirica sviluppa l'intuizione sul raggiungimento di una quantizzazione ottimale.

Gli strumenti di visualizzazione possono fornire approfondimenti che aiutano i professionisti a comprendere meglio gli effetti delle ottimizzazioni sui loro modelli. La visibilità consente di correggere i problemi in anticipo prima che l'accuratezza o le prestazioni siano influenzate in modo significativo. Aiuta anche ad applicare le ottimizzazioni in modo più efficace per modelli specifici. Queste analisi di ottimizzazione aiutano a sviluppare l'intuizione quando si trasferiscono i modelli a rappresentazioni più efficienti.

9.5.5 Conversione e Distribuzione del Modello

Una volta che i modelli sono stati ottimizzati con successo in framework come TensorFlow e PyTorch, sono necessarie piattaforme specializzate di conversione e deployment distribuzione del modello per colmare il divario con l'esecuzione sui dispositivi target.

TensorFlow Lite - La piattaforma di TensorFlow per convertire i modelli in un formato leggero ottimizzato per dispositivi mobili, embedded ed edge. Supporta ottimizzazioni come quantizzazione, fusione del kernel e rimozione di operazioni inutilizzate. I modelli possono essere eseguiti utilizzando kernel TensorFlow Lite ottimizzati sull'hardware del dispositivo. Fondamentale per la distribuzione mobile e TinyML.

ONNX Runtime - Esegue la conversione e l'inferenza per i modelli nel formato "open ONNX". Fornisce kernel ottimizzati, supporta acceleratori hardware come GPU e distribuzione multipiattaforma dal cloud all'edge. Consente la distribuzione indipendente dal framework. Figura 9.42 è una mappa di interoperabilità ONNX, inclusi i principali framework più diffusi.

Figura 9.42: Interoperabilità di ONNX. Fonte: TowardsDataScience.

PyTorch Mobile - Consente l'esecuzione dei modelli PyTorch su iOS e Android convertendoli in rappresentazioni ottimizzate per dispositivi mobili. Fornisce implementazioni mobili efficienti di operazioni come convoluzione e funzioni speciali ottimizzate per hardware mobile.

Queste piattaforme si integrano con driver hardware, sistemi operativi e librerie di acceleratori sui dispositivi per eseguire modelli in modo efficiente utilizzando l'ottimizzazione hardware. Inoltre, delegano le operazioni ad acceleratori ML dedicati, ove presenti. La disponibilità di queste piattaforme di distribuzione collaudate e robuste colma il divario tra l'ottimizzazione dei modelli nei framework e la distribuzione effettiva su miliardi di dispositivi. Consentono agli utenti di concentrarsi sullo sviluppo del modello anziché sulla creazione di runtime mobili personalizzati. L'innovazione continua per supportare nuovi hardware e ottimizzazioni in queste piattaforme è fondamentale per le ottimizzazioni di ML diffuse. Fornendo queste pipeline di distribuzione ottimizzate, l'intero flusso di lavoro, dal training al deployment distribuzione del dispositivo, può sfruttare le ottimizzazioni del modello per fornire applicazioni ML performanti. Questa infrastruttura software end-to-end ha contribuito a guidare l'adozione di ML sul dispositivo.

9.6 Conclusione

In questo capitolo abbiamo discusso l'ottimizzazione del modello nell'ambito software-hardware. Ci siamo immersi in una rappresentazione efficiente del modello, dove abbiamo trattato le sfumature della potatura strutturata e non-strutturata e altre tecniche per la compressione del modello come la distillazione della conoscenza e la decomposizione di matrice e tensore. Ci siamo anche immersi brevemente nella progettazione del modello specifico per l'edge a livello di parametri e architettura del modello, esplorando argomenti come modelli specifici per l'edge e NAS basati sull'hardware.

Abbiamo quindi esplorato rappresentazioni numeriche efficienti, dove abbiamo trattato le basi della matematica, codifiche numeriche e archiviazione, vantaggi della matematica efficiente e le sfumature della rappresentazione numerica con utilizzo della memoria, complessità computazionale, compatibilità hardware e scenari di compromesso. Abbiamo concluso concentrandoci su un elemento fondamentale della matematica efficiente: la

quantizzazione, dove abbiamo esaminato la sua storia, calibrazione, tecniche e interazione con la potatura.
Infine, abbiamo esaminato come possiamo apportare ottimizzazioni specifiche per l'hardware che abbiamo. Abbiamo esplorato come possiamo trovare architetture modello su misura per l'hardware, apportare ottimizzazioni nel kernel per gestire meglio il modello e framework creati per sfruttare al meglio l'hardware. Abbiamo anche esaminato come possiamo fare il contrario e creare hardware attorno al nostro software specifico e abbiamo parlato di come suddividere le reti per l'esecuzione su più processori disponibili sul dispositivo edge.
Comprendendo il quadro completo dei gradi di libertà all'interno dell'ottimizzazione del modello sia lontano che vicino all'hardware e i compromessi da considerare quando si implementano questi metodi, i professionisti possono sviluppare una pipeline più ponderata per comprimere i loro carichi di lavoro sui dispositivi edge.

9.7 Risorse

Ecco un elenco curato di risorse per supportare sia gli studenti che gli insegnanti nel loro percorso di apprendimento e insegnamento. Stiamo lavorando costantemente per espandere questa raccolta e aggiungeremo nuovi esercizi nel prossimo futuro.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale al proprio ritmo. Incoraggiamo sia gli studenti che gli insegnanti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Quantizzazione:
  - Quantization: Part 1.
  - Quantization: Part 2.
  - Post-Training Quantization (PTQ).
  - Quantization-Aware Training (QAT).
- Pruning:
  - Pruning: Part 1.
  - Pruning: Part 2.
- Knowledge Distillation.
- Clustering.
- Neural Architecture Search (NAS):
  - NAS overview.
  - NAS: Part 1.
  - NAS: Part 2.

Video

- Video 9.1

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 9.1
- Esercizio 9.2
- Esercizio 9.3

Laboratori

Oltre agli esercizi, offriamo anche una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

10. Accelerazione IA

DALL·E 3 Prompt: Creare una rappresentazione intricata e colorata di un progetto di System on Chip (SoC) in un formato rettangolare. Mostrare una varietà di acceleratori e piccoli chip di machine learning specializzati, tutti integrati nel processore. Fornire una vista dettagliata all'interno del chip, evidenziando il rapido movimento degli elettroni. Ogni acceleratore e piccolo chip dovrebbe essere progettato per interagire con i neuroni, gli strati e le attivazioni della rete neurale, enfatizzandone la velocità di elaborazione. Rappresentare le reti neurali come una rete di nodi interconnessi, con flussi di dati vibranti che scorrono tra i pezzi dell'acceleratore, mostrando la migliorata velocità di elaborazione.

L'implementazione di ML su dispositivi edge presenta sfide quali velocità di elaborazione limitata, vincoli di memoria e rigorosi requisiti di efficienza energetica. Per superare queste sfide, l'accelerazione hardware specializzata è fondamentale. Gli acceleratori hardware sono progettati per ottimizzare attività ad alta intensità di calcolo come l'inferenza utilizzando chip di silicio personalizzati su misura per moltiplicazioni di matrici, fornendo accelerazioni significative rispetto alle CPU per uso generico. Ciò consente l'esecuzione in tempo reale di modelli avanzati su dispositivi con rigorosi vincoli di dimensioni, peso e potenza.

Obiettivi dell'Apprendimento

- Comprendere perché l'accelerazione hardware è necessaria per i carichi di lavoro AI
- Esaminare le opzioni chiave di accelerazione come GPU, TPU, FPGA e ASIC e i loro compromessi
- Scoprire modelli di programmazione, framework e compilatori per acceleratori AI
- Apprezzare l'importanza del benchmarking e delle metriche per la valutazione hardware
- Riconoscere il ruolo della progettazione congiunta hardware-software nella creazione di sistemi efficienti
- Ottenere visibilità su direzioni di ricerca all'avanguardia come il calcolo neuromorfico e quantistico
- Comprendere come il ML sta iniziando ad aumentare e migliorare la progettazione hardware

10.1 Introduzione

Probabilmente avrete notato la crescente domanda di integrazione dell'apprendimento automatico nei dispositivi di uso quotidiano, come gli smartphone nelle nostre tasche, gli elettrodomestici intelligenti e persino i veicoli autonomi. Portare le funzionalità di ML in questi ambienti del mondo reale è entusiasmante, ma comporta una serie di sfide. A differenza dei potenti server dei data center, questi dispositivi edge hanno risorse di elaborazione limitate, il che rende difficile eseguire modelli complessi in modo efficace.
L'accelerazione hardware specializzata è la chiave per rendere possibile l'apprendimento automatico ad alte prestazioni su dispositivi edge con risorse limitate. Quando parliamo di accelerazione hardware, ci riferiamo all'uso di chip e architetture personalizzati progettati per gestire il pesante lavoro delle operazioni di ML, alleggerendo il carico del processore principale. Nelle reti neurali, alcune delle attività più impegnative riguardano le moltiplicazioni di matrici durante l'inferenza. Gli acceleratori hardware sono progettati per ottimizzare queste operazioni, spesso offrendo accelerazioni da 10 a 100 volte superiori rispetto alle CPU per uso generico. Questo tipo di accelerazione è ciò che rende fattibile l'esecuzione di modelli di reti neurali avanzate su dispositivi limitati da dimensioni, peso e potenza, e di fare tutto in tempo reale.
In questo capitolo, esamineremo più da vicino le diverse tecniche di accelerazione hardware disponibili per l'apprendimento automatico embedded e i compromessi che derivano da ciascuna opzione. L'obiettivo è fornire una solida comprensione di come funzionano queste tecniche, in modo che si possano prendere decisioni informate quando si tratta di scegliere l'hardware giusto e ottimizzare il software. Alla fine, sarete ben equipaggiati per sviluppare capacità di apprendimento automatico ad alte prestazioni su dispositivi edge, anche con i loro vincoli.

10.2 Background e Basi

10.2.1 Background Storico

Le origini dell'accelerazione hardware risalgono agli anni '60, con l'avvento dei coprocessori matematici in virgola mobile per eliminare i calcoli dalla CPU principale. Un primo esempio è stato il chip Intel 8087 rilasciato nel 1980 per accelerare le operazioni in virgola mobile per il processore 8086. Ciò ha stabilito la pratica di utilizzare processori specializzati per gestire in modo efficiente carichi di lavoro ad alta intensità di calcolo.
Negli anni '90, sono emerse le prime Graphics Processing Units (GPU) [Unità di elaborazione grafica] per elaborare rapidamente pipeline grafiche per rendering e giochi. La GeForce 256 di Nvidia nel 1999 è stata una delle prime GPU programmabili in grado di eseguire algoritmi software personalizzati. Le GPU esemplificano acceleratori a funzione fissa specifici per dominio e si sono evolute in acceleratori programmabili paralleli.
Negli anni 2000, le GPU sono state applicate all'elaborazione generica in GPGPU. La loro elevata larghezza di banda di memoria e la produttività computazionale le hanno rese adatte a carichi di lavoro ad alta intensità di calcolo. Ciò ha incluso innovazioni nell'uso di GPU per accelerare il training di modelli di deep learning come AlexNet nel 2012.
Negli ultimi anni, le Tensor Processing Unit (TPU) di Google rappresentano ASIC personalizzati specificamente progettati per la moltiplicazione di matrici nel deep learning. Durante l'inferenza, i loro core tensoriali ottimizzati raggiungono TeraOPS/watt più elevati rispetto a CPU o GPU. L'innovazione continua include tecniche di compressione del modello come pruning e quantizzazione per adattare reti neurali più grandi su dispositivi edge.
Questa evoluzione dimostra come l'accelerazione hardware si sia concentrata sulla risoluzione di colli di bottiglia ad alta intensità di calcolo, dalla matematica in virgola mobile

alla grafica alla moltiplicazione di matrici per ML. Comprendere questa storia fornisce un contesto cruciale per gli acceleratori AI specializzati odierni.

10.2.2 La Necessità di Accelerazione

L'evoluzione dell'accelerazione hardware è strettamente legata alla storia più ampia dell'informatica. Centrale in questa storia è il ruolo dei transistor, i mattoni fondamentali dell'elettronica moderna. I transistor agiscono come piccoli interruttori che possono accendersi o spegnersi, consentendo i calcoli complessi che guidano tutto, dalle semplici calcolatrici ai modelli avanzati di apprendimento automatico. Nei primi decenni, la progettazione dei chip era governata dalla legge di Moore, che prevedeva che il numero di transistor su un circuito integrato sarebbe raddoppiato approssimativamente ogni due anni, e dal Dennard Scaling, che osservava che man mano che i transistor diventavano più piccoli, le loro prestazioni (velocità) aumentavano, mentre la densità di potenza (potenza per unità di area) rimaneva costante. Queste due leggi sono state mantenute durante l'era single-core. Figura 10.1 mostra le tendenze di diverse metriche dei microprocessori. Come indica la figura, il Dennard Scaling fallisce intorno alla metà degli anni 2000; si noti come la velocità di clock (frequenza) rimanga pressoché costante anche se il numero di transistor continua ad aumentare.

Figura 10.1: Tendenze dei Microprocessori. Fonte: Karl Rupp.

Tuttavia, come descrive D. A. Patterson e Hennessy (2016), i vincoli tecnologici alla fine hanno imposto una transizione all'era multicore, con chip contenenti più core di elaborazione per offrire guadagni in termini di prestazioni. Le limitazioni di potenza hanno impedito un ulteriore ridimensionamento, il che ha portato al "silicio scuro" (Dark Silicon), in cui non tutte le aree del chip potevano essere attive simultaneamente (Xiu 2019). "Dark silicon" si riferisce a parti del chip che non possono essere alimentate simultaneamente a causa di limitazioni termiche e di potenza. In sostanza, con l'aumento della densità dei transistor, la quota del chip che poteva essere utilizzata attivamente senza surriscaldarsi o superare i budget di potenza si è ridotta.

Questo fenomeno ha comportato che, sebbene i chip avessero più transistor, non tutti potevano essere operativi simultaneamente, limitando i potenziali guadagni in termini di prestazioni. Questa crisi energetica ha reso necessario un passaggio all'era degli acceleratori, con unità hardware specializzate su misura per attività specifiche per massimizzare l'efficienza. L'esplosione dei carichi di lavoro dell'intelligenza artificiale ha ulteriormente spinto la domanda di acceleratori personalizzati. I fattori abilitanti includevano nuovi linguaggi di programmazione, strumenti software e progressi nella produzione. Fondamentalmente, gli acceleratori hardware vengono valutati in base a Prestazioni, Potenza e Area di silicio (PPA); la natura dell'applicazione target, sia essa legata alla memoria o al calcolo, influenza notevolmente la progettazione. Ad esempio, i carichi di lavoro legati alla memoria richiedono un'elevata larghezza di banda e un accesso a bassa latenza, mentre le applicazioni legate al calcolo richiedono la massima produttività di elaborazione.

10.2.3 Principi Generali

La progettazione di acceleratori hardware specializzati comporta la gestione di compromessi complessi tra prestazioni, efficienza energetica, area di silicio e ottimizzazioni specifiche del carico di lavoro. Questa sezione delinea considerazioni e metodologie fondamentali per raggiungere un equilibrio ottimale in base ai requisiti dell'applicazione e ai vincoli hardware.

10.2.3.1 Prestazioni entro i Budget di Potenza

Per capire come raggiungere il giusto equilibrio tra prestazioni e budget di potenza, è importante definire prima alcuni concetti chiave che svolgono un ruolo cruciale in questo processo. Le prestazioni si riferiscono in generale alla capacità complessiva di un sistema di completare efficacemente le attività di calcolo entro determinati vincoli. Uno dei componenti chiave delle prestazioni è il throughput, ovvero la velocità con cui vengono elaborate queste attività, comunemente misurata in "floating point operations per second (FLOPS)" [operazioni in virgola mobile al secondo] o frame al secondo (FPS). Il throughput dipende fortemente dal parallelismo, ovvero la capacità dell'hardware di eseguire più operazioni contemporaneamente, e dalla frequenza di clock, ovvero la velocità con cui il processore esegue ciclicamente queste operazioni. Un throughput più elevato in genere comporta prestazioni migliori, ma aumenta anche il consumo di energia all'aumentare dell'attività.
La semplice massimizzazione del throughput non è sufficiente; anche l'efficienza dell'hardware è importante. L'efficienza è la misura di quante operazioni vengono eseguite per watt di potenza consumata, riflettendo la relazione tra lavoro di calcolo e consumo di energia. In scenari in cui la potenza è un fattore limitante, come nei dispositivi edge, ottenere un'elevata efficienza è fondamentale. Per aiutare a ricordare come questi concetti si interconnettono, considerare le seguenti relazioni:

- **Prestazioni** = Throughput * Efficienza
- **Throughput** ~= Parallelismo * Frequenza di Clock
- **Efficienza** = Operazioni / Watt

Gli acceleratori hardware mirano a massimizzare le prestazioni entro budget di potenza stabiliti. Ciò richiede un attento bilanciamento del parallelismo, della frequenza di clock del chip, della tensione di esercizio, dell'ottimizzazione del carico di lavoro e di altre tecniche per massimizzare le operazioni per watt.
Ad esempio, le GPU raggiungono un throughput elevato tramite architetture massivamente parallele. Tuttavia, la loro efficienza è inferiore a quella dei circuiti integrati specifici per applicazione (ASIC) personalizzati come il TPU di Google, che ottimizzano per un carico di lavoro specifico.

10.2.3.2 Gestione dell'Area e dei Costi del Silicio

La dimensione dell'area di un chip ha un impatto diretto sul suo costo di produzione. Per capirne il motivo, è utile conoscere un po' il processo di produzione.
I chip vengono creati da grandi e sottili fette di materiale semiconduttore note come wafer. Durante la produzione, ogni wafer viene suddiviso in blocchi più piccoli chiamati "die", e ogni die contenente i circuiti per un singolo chip. Dopo che il wafer è stato elaborato, viene tagliato in questi singoli die, che vengono poi confezionati per formare i chip finali utilizzati nei dispositivi elettronici.
I die più grandi richiedono più materiale e sono più inclini a difetti, il che può ridurre la resa, il che significa che vengono prodotti meno chip utilizzabili da ogni wafer. Mentre i produttori possono scalare i progetti combinando più die più piccoli in un singolo pacchetto (pacchetti multi-die), ciò aggiunge complessità e costi al processo di confezionamento e produzione.
La quantità di area di silicio necessaria su un die dipende da diversi fattori:

- **Risorse di Calcolo**, ad esempio numero di core, memoria, cache
- **Nodo del Processo di Produzione**, transistor più piccoli consentono una maggiore densità
- **Modello di Programmazione**, acceleratori programmati richiedono maggiore flessibilità

La progettazione dell'acceleratore implica la compressione delle massime prestazioni entro questi vincoli di area del silicio. Tecniche come la potatura e la compressione aiutano ad adattare modelli più grandi al chip senza superare lo spazio disponibile.

10.2.3.3 Ottimizzazioni Specifiche del Carico di Lavoro

La progettazione di acceleratori hardware efficaci richiede di adattare l'architettura alle esigenze specifiche del carico di lavoro target. Diversi tipi di carichi di lavoro, che siano in AI, grafica o robotica, hanno caratteristiche uniche che stabiliscono come l'acceleratore dovrebbe essere ottimizzato.
Alcune delle considerazioni chiave quando si ottimizza l'hardware per carichi di lavoro specifici includono:

- **Memoria vs Limiti di Calcolo:** I carichi di lavoro vincolati alla memoria richiedono una maggiore larghezza di banda di memoria, mentre le app vincolate al calcolo necessitano di un throughput [produttività] aritmetico.
- **Località dei Dati:** Lo spostamento dei dati dovrebbe essere ridotto al minimo per l'efficienza. La memoria vicina al calcolo aiuta.
- **Operazioni a Livello di Bit:** I tipi di dati a bassa precisione come INT8/INT4 ottimizzano la densità di calcolo.
- **Parallelismo dei Dati:** Più unità di calcolo replicate consentono l'esecuzione parallela.
- **Pipelining:** L'esecuzione sovrapposta delle operazioni aumenta la produttività.

La comprensione delle caratteristiche del carico di lavoro consente un'accelerazione personalizzata. Ad esempio, le reti neurali convoluzionali utilizzano operazioni di "finestra scorrevole" mappate in modo ottimale su array spaziali di elementi di elaborazione.
Grazie alla comprensione di questi compromessi architettonici, i progettisti possono prendere decisioni informate sull'architettura dell'acceleratore hardware, assicurandosi che fornisca le migliori prestazioni possibili per l'uso previsto.

10.2.3.4 Progettazione Hardware Sostenibile

Negli ultimi anni, la sostenibilità dell'IA è diventata una preoccupazione urgente, guidata da due fattori chiave: la scala crescente dei carichi di lavoro dell'IA e il consumo energetico associato.

Innanzitutto, le dimensioni dei modelli e dei set di dati dell'IA sono cresciute rapidamente. Ad esempio, in base alle tendenze di elaborazione dell'IA di OpenAI, la quantità di elaborazione utilizzata per addestrare modelli all'avanguardia raddoppia ogni 3,5 mesi. Questa crescita esponenziale richiede enormi risorse di elaborazione nei data center.

In secondo luogo, l'uso di energia per l'addestramento e l'inferenza dell'IA presenta problemi di sostenibilità. I data center che eseguono applicazioni di IA consumano molta energia, contribuendo a elevate emissioni di carbonio. Si stima che l'addestramento di un grande modello di IA possa avere un'impronta di carbonio di 626.000 libbre di CO_2 equivalente, quasi 5 volte le emissioni di un'auto media nel corso della sua vita.

Per affrontare queste sfide, la progettazione hardware sostenibile si concentra sull'ottimizzazione dell'efficienza energetica senza compromettere le prestazioni. Ciò comporta lo sviluppo di acceleratori specializzati che riducono al minimo il consumo di energia massimizzando al contempo la produttività computazionale.

Parleremo di IA sostenibile in un capitolo successivo, dove ne discuteremo più in dettaglio.

10.3 Tipi di acceleratori

Gli acceleratori hardware possono assumere molte forme. Possono esistere come widget (come il Neural Engine nel chip Apple M1) o come interi chip appositamente progettati per svolgere molto bene determinate attività. Questa sezione esaminerà i processori per carichi di lavoro di apprendimento automatico lungo lo spettro che va dagli ASIC altamente specializzati alle CPU più generiche.

Ci concentriamo prima sull'hardware personalizzato appositamente progettato per l'intelligenza artificiale per comprendere le ottimizzazioni più estreme possibili quando vengono rimossi i vincoli di progettazione. Questo stabilisce un limite massimo per prestazioni ed efficienza. Poi prendiamo in considerazione progressivamente architetture più programmabili e adattabili, discutendo di GPU e FPGA. Queste fanno compromessi nella personalizzazione per mantenere la flessibilità. Infine, trattiamo le CPU generiche che sacrificano le ottimizzazioni per un carico di lavoro particolare in cambio di una programmabilità versatile tra le applicazioni.

Strutturando l'analisi lungo questo spettro, miriamo a illustrare i compromessi fondamentali tra utilizzo, efficienza, programmabilità e flessibilità nella progettazione dell'acceleratore. Il punto di equilibrio ottimale dipende dai vincoli e dai requisiti dell'applicazione target. Questa prospettiva dello spettro fornisce un quadro per ragionare sulle scelte hardware per l'apprendimento automatico e sulle capacità richieste a ciascun livello di specializzazione.

Figura 10.2 illustra la complessa interazione tra flessibilità, prestazioni, diversità funzionale e area di progettazione dell'architettura. Notare come l'ASIC si trovi nell'angolo in basso a destra, con area minima, flessibilità e consumo energetico e prestazioni massime, a causa della sua natura altamente specializzata per l'applicazione. Un compromesso chiave è la diversità funzionale rispetto alle prestazioni: le architetture per uso generico possono servire applicazioni diverse, ma le loro prestazioni applicative sono degradate rispetto alle architetture più personalizzate.

Figura 10.2: Compromessi di Progettazione. Fonte: El-Rayis (2014).

La progressione inizia con l'opzione più specializzata, gli ASIC appositamente progettati per l'intelligenza artificiale, per basare la nostra comprensione sulle massime ottimizzazioni possibili prima di espanderci ad architetture più generalizzabili. Questo approccio strutturato chiarisce lo spazio di progettazione dell'acceleratore.

10.3.1 Application-Specific Integrated Circuits (ASIC)

Un "circuito integrato specifico per applicazione" (ASIC) è un tipo di circuito integrato (IC) progettato su misura per un'applicazione o un carico di lavoro specifico, anziché per un uso generico. A differenza di CPU e GPU, gli ASIC non supportano più applicazioni o carichi di lavoro. Piuttosto, sono ottimizzati per eseguire un singolo compito in modo estremamente efficiente. Google TPU è un esempio di ASIC.

Gli ASIC raggiungono questa efficienza adattando ogni aspetto del design del chip, ovvero le porte logiche sottostanti, i componenti elettronici, l'architettura, la memoria, l'I/O e il processo di produzione, specificamente per l'applicazione target. Questo livello di personalizzazione consente di rimuovere qualsiasi logica o funzionalità non necessaria richiesta per il calcolo generale. Il risultato è un IC che massimizza le prestazioni e l'efficienza energetica sul carico di lavoro desiderato. I guadagni di efficienza derivanti dall'hardware specifico per applicazione sono così sostanziali che queste aziende incentrate sul software dedicano enormi risorse ingegneristiche alla progettazione di ASIC personalizzati.

L'ascesa di algoritmi di apprendimento automatico più complessi ha reso i vantaggi prestazionali abilitati dall'accelerazione hardware personalizzata un fattore di differenziazione competitiva chiave, anche per le aziende tradizionalmente concentrate sull'ingegneria del software. Gli ASIC sono diventati un investimento ad alta priorità per i principali provider cloud che mirano a offrire un calcolo AI più veloce.

10.3.1.1 Vantaggi

Grazie alla loro natura personalizzata, gli ASIC offrono vantaggi significativi rispetto ai processori generici come CPU e GPU. I principali vantaggi includono quanto segue.

10.3.1.1.1 Prestazioni ed efficienza massimizzate

Il vantaggio più fondamentale degli ASIC è la massimizzazione delle prestazioni e dell'efficienza energetica personalizzando l'architettura hardware specificamente per l'applicazione target. Ogni transistor e aspetto della progettazione è ottimizzato per il carico

di lavoro desiderato: non è necessaria alcuna logica o sovraccarico non necessario per supportare il calcolo generico.
Ad esempio, le Tensor Processing Units (TPU) di Google contengono architetture su misura esattamente per le operazioni di moltiplicazione di matrici utilizzate nelle reti neurali. Per progettare gli ASIC TPU, i team di ingegneria di Google devono definire chiaramente le specifiche del chip, scrivere la descrizione dell'architettura utilizzando linguaggi di descrizione hardware come Verilog, sintetizzare il design per mapparlo sui componenti hardware e posizionare e instradare con cura transistor e collegamenti in base alle regole di progettazione del processo di fabbricazione. Questo complesso processo di progettazione, noto come "very-large-scale integration" (VLSI) [integrazione su larga scala], consente loro di creare un IC ottimizzato per carichi di lavoro di apprendimento automatico.
Di conseguenza, gli ASIC TPU raggiungono un'efficienza di oltre un ordine di grandezza superiore nelle operazioni per watt rispetto alle GPU per uso generico sui carichi di lavoro di apprendimento automatico massimizzando le prestazioni e riducendo al minimo il consumo energetico tramite un design hardware full-stack personalizzato.

10.3.1.1.2 Memoria On-Chip Specializzata

Gli ASIC incorporano memoria on-chip, come SRAM (Static Random Access Memory) e cache specificamente ottimizzate per fornire dati alle unità di elaborazione. La SRAM è un tipo di memoria più veloce e affidabile della DRAM (Dynamic Random Access Memory) perché non deve essere aggiornata periodicamente. Tuttavia, richiede più transistor per bit di dati, il che la rende più ingombrante e costosa da produrre rispetto alla DRAM.
La SRAM è ideale per la memoria on-chip, dove la velocità è fondamentale. Il vantaggio di avere grandi quantità di SRAM on-chip ad alta larghezza di banda è che i dati possono essere archiviati vicino agli elementi di elaborazione, consentendo un rapido accesso. Ciò fornisce enormi vantaggi in termini di velocità rispetto all'accesso alla DRAM off-chip, che, sebbene di capacità maggiore, può essere fino a 100 volte più lenta. Ad esempio, il system-on-a-chip M1 di Apple contiene una speciale SRAM a bassa latenza per accelerare le prestazioni del suo hardware di machine learning Neural Engine.
La località dei dati e l'ottimizzazione della gerarchia di memoria sono fondamentali per un throughput elevato e un basso consumo energetico. Tabella 10.1 mostra "Numeri che Tutti Dovrebbero Conoscere", di Jeff Dean.

Tabella 10.1: Confronto della latenza delle operazioni di elaborazione e di rete.

Operazione	Latenza
Riferimento alla cache L1	0,5 ns
Branch mispredict	5 ns
Riferimento alla cache L2	7 ns
Blocco/sblocco mutex	25 ns
Riferimento alla memoria principale	100 ns
Comprimere 1K byte con Zippy	3.000 ns (3 us)
Inviare 1 KB byte su una rete da 1 Gbps	10.000 ns (10 us)
Leggere 4 KB casualmente da SSD	150.000 ns (150 us)
Leggere 1 MB in sequenza dalla memoria	250.000 ns (250 us)
Andata e ritorno all'interno dello stesso data center	500.000 ns (0,5 ms)
Leggere 1 MB in sequenza da SSD	1.000.000 ns (1 ms)
Ricerca su disco	10.000.000 ns (10 ms)
Leggere 1 MB in sequenza da disco	20.000.000 ns (20 ms)
Inviare un pacchetto CA → Paesi Bassi → CA	150.000.000 ns (150 ms)

10.3.1.1.3 Tipi di Dati e Operazioni Personalizzati

A differenza dei processori generici, gli ASIC possono essere progettati per supportare in modo nativo tipi di dati personalizzati come INT4 o bfloat16, ampiamente utilizzati nei modelli di ML. Ad esempio, l'architettura GPU Ampere di Nvidia ha un bfloat16 dedicato ai Tensor Core per accelerare i carichi di lavoro AI. I tipi di dati a bassa precisione consentono una maggiore densità aritmetica e prestazioni. Per ulteriori dettagli fare riferimento a Sezione 8.6. Gli ASIC possono anche incorporare direttamente operazioni non standard negli algoritmi ML come operazioni primitive, ad esempio, il supporto nativo di funzioni di attivazione come ReLU rende l'esecuzione più efficiente.

10.3.1.1.4 Parallelismo Elevato

Le architetture ASIC possono sfruttare un parallelismo più elevato ottimizzato per il carico di lavoro del target rispetto alle CPU o GPU generiche. Un maggior numero di unità di calcolo personalizzate per l'applicazione significa più operazioni eseguite simultaneamente. Gli ASIC altamente paralleli raggiungono un throughput enorme per carichi di lavoro paralleli di dati come l'inferenza di reti neurali.

10.3.1.1.5 Nodi di Processo Avanzati

I processi di produzione all'avanguardia consentono di impacchettare più transistor in aree di die più piccole, aumentando la densità. Gli ASIC progettati specificamente per applicazioni ad alto volume possono ammortizzare meglio i costi dei nodi.

10.3.1.2 Svantaggi

10.3.1.2.1 Tempistiche di Progettazione Lunghe

Il processo di progettazione e validazione di un ASIC può richiedere 2-3 anni. La sintesi dell'architettura utilizzando linguaggi di descrizione hardware, la definizione del layout del chip e la fabbricazione del chip su nodi di processo avanzati comportano lunghi cicli di sviluppo. Ad esempio, per realizzare un chip da 7 nm, i team devono definire attentamente le specifiche, scrivere l'architettura in HDL, sintetizzare le porte logiche, posizionare i componenti, instradare tutte le interconnessioni e finalizzare il layout da inviare per la fabbricazione. Questa "Very Large-Scale Integration (VLSI)" significa che la progettazione e la produzione di ASIC possono tradizionalmente richiedere 2-5 anni.
Ci sono alcuni motivi chiave per cui le lunghe tempistiche di progettazione degli ASIC, spesso 2-3 anni, possono essere difficili per i carichi di lavoro di apprendimento automatico:

- **Gli algoritmi ML si evolvono rapidamente:** Nuove architetture di modelli, tecniche di training e ottimizzazioni di rete emergono continuamente. Ad esempio, i Transformers sono diventati estremamente popolari nell'NLP negli ultimi anni. Quando un ASIC termina il tapeout, l'architettura ottimale per un carico di lavoro potrebbe essere cambiata.
- **I dataset crescono rapidamente:** Gli ASIC progettati per determinate dimensioni di modello o tipi di dati possono diventare sottodimensionati rispetto alla domanda. Ad esempio, i modelli di linguaggio naturale stanno aumentando esponenzialmente con più dati e parametri. Un chip progettato per BERT potrebbe non supportare GPT-3.
- **Le applicazioni ML cambiano frequentemente:** L'attenzione del settore cambia tra visione artificiale, parlato, NLP, sistemi di raccomandazione, ecc. Un ASIC ottimizzato per la classificazione delle immagini potrebbe avere meno rilevanza in pochi anni.
- **Cicli di progettazione più rapidi con GPU/FPGA:** Gli acceleratori programmabili come le GPU possono adattarsi molto più rapidamente aggiornando le librerie software e i framework. I nuovi algoritmi possono essere implementati senza modifiche hardware.

- **Esigenze di time-to-market:** Ottenere un vantaggio competitivo in ML richiede di sperimentare e implementare rapidamente nuove idee. Attendere diversi anni per un ASIC è diverso da un'iterazione rapida.

Il ritmo dell'innovazione in ML deve essere adattato meglio alla scala temporale pluriennale per lo sviluppo di ASIC. Sono necessari notevoli sforzi ingegneristici per estendere la durata di vita di ASIC tramite architetture modulari, ridimensionamento dei processi, compressione dei modelli e altre tecniche. Tuttavia, la rapida evoluzione di ML rende l'hardware a funzione fissa una sfida.

10.3.1.2.2 Elevati Costi di Progettazione Non Ricorrenti

I costi fissi per portare un ASIC dalla progettazione alla produzione ad alto volume possono essere molto dispendiosi in termini di capitale, spesso decine di milioni di dollari. La fabbricazione di fotomaschere per il tape-out dei chip in nodi di processo avanzati, il packaging e il lavoro di progettazione una tantum sono costosi. Ad esempio, un solo tape-out del chip da 7 nm potrebbe costare milioni. L'elevato "non-recurring engineering (NRE)" [investimento di progettazione non ricorrente] riduce la fattibilità dell'ASIC ai casi di utilizzo della produzione ad alto volume in cui il costo iniziale può essere ammortizzato.

10.3.1.2.3 Integrazione e Programmazione Complesse

Gli ASIC richiedono un ampio lavoro di integrazione software, inclusi driver, compilatori, supporto del sistema operativo e strumenti di debug. Hanno anche bisogno di competenza nel packaging elettrico e termico. Inoltre, programmare in modo efficiente le architetture ASIC può comportare sfide come il partizionamento del carico di lavoro e la pianificazione su molte unità parallele. La natura personalizzata richiede notevoli sforzi di integrazione per trasformare l'hardware grezzo in acceleratori completamente operativi.
Mentre gli ASIC forniscono enormi guadagni di efficienza nelle applicazioni target adattando ogni aspetto della progettazione hardware a un'attività specifica, la loro natura fissa comporta compromessi in termini di flessibilità e costi di sviluppo rispetto agli acceleratori programmabili, che devono essere soppesati in base all'applicazione.

10.3.2 Field-Programmable Gate Array (FPGA)

Gli FPGA sono circuiti integrati programmabili che possono essere riconfigurati per diverse applicazioni. La loro natura personalizzabile offre vantaggi per accelerare gli algoritmi AI rispetto agli ASIC fissi o alle GPU inflessibili. Mentre Google, Meta e NVIDIA stanno valutando di installare gli ASIC nei data center, Microsoft ha distribuito gli FPGA nei suoi data center (Putnam et al. 2014) nel 2011 per servire in modo efficiente diversi carichi di lavoro.
Gli FPGA hanno trovato ampia applicazione in vari campi, tra cui l'imaging medico, la robotica e la finanza, dove eccellono nella gestione di attività di machine learning ad alta intensità di calcolo. Nell'imaging medico, un esempio illustrativo è l'applicazione degli FPGA per la segmentazione dei tumori cerebrali, un processo tradizionalmente dispendioso in termini di tempo e soggetto a errori. Rispetto alle implementazioni tradizionali di GPU e CPU, gli FPGA hanno dimostrato rispettivamente miglioramenti delle prestazioni di oltre 5 e 44 volte e guadagni di 11 e 82 volte in termini di efficienza energetica, evidenziando il loro potenziale per applicazioni esigenti (Xiong et al. 2021).

10.3.2.1 Vantaggi

Gli FPGA offrono diversi vantaggi rispetto alle GPU e agli ASIC per accelerare i carichi di lavoro di apprendimento automatico.

10.3.2.1.1 Flessibilità Tramite "Reconfigurable Fabric"

Il vantaggio principale degli FPGA è la capacità di riconfigurare il "fabric" [tessuto] sottostante per implementare architetture personalizzate ottimizzate per diversi modelli, a

differenza degli ASIC a funzione fissa. Ad esempio, le società di trading quantitativo utilizzano gli FPGA per accelerare i loro algoritmi perché cambiano frequentemente e il basso costo NRE degli FPGA è più fattibile rispetto acquistare i nuovi ASIC. Figura 10.3 contiene una tabella che confronta tre diversi FPGA.

	Lattice Certus-NX-17	Intel Cyclone V 5CEA2	Xilinx Spartan-7 XC7S25
Logic Cells	17,000 LCs	25,000 LEs	23,360 LCs
Total RAM	3.0Mbits	1.9Mbits	1.9Mbits
DSP (18x18 Mult)	24 multipliers	50 multipliers	80 multipliers
Hard Crypto Blocks	AES, ECDSA	AES	AES
Other Hard I/O	SGMII, ADC	DDR3	ADC
Max I/O Pins	78 I/O	223 GPIO	150 I/O
Max I/O Density	2.2 per mm^2	1.3 per mm^2	0.9 per mm^2
Diff I/O Speed	1,500Mbps	840Mbps	1,250Mbps
IC Process	28nm FD-SOI	28nm CMOS	28nm CMOS
Min Package Size	6mm x 6mm	13mm x 13mm	13mm x 13mm

Figura 10.3: Confronto di FPGA. Fonte: Gwennap (s.d.).

Gli FPGA comprendono blocchi di base: blocchi logici configurabili, blocchi RAM e interconnessioni. Vengono fornite una quantità base di queste risorse e gli ingegneri programmano i chip compilando il codice HDL in flussi di bit che riorganizzano la struttura in diverse configurazioni. Questo rende gli FPGA adattabili man mano che gli algoritmi evolvono. Sebbene gli FPGA possano non raggiungere le massime prestazioni ed efficienza degli ASIC specifici per il carico di lavoro, la loro programmabilità offre maggiore flessibilità man mano che gli algoritmi cambiano. Questa adattabilità rende gli FPGA una scelta interessante per accelerare le applicazioni di machine learning in evoluzione.

10.3.2.1.2 Parallelismo e Pipeline Personalizzati

Le architetture FPGA possono sfruttare il parallelismo spaziale e il pipelining adattando la progettazione hardware per rispecchiare il parallelismo nei modelli ML. Ad esempio, su una piattaforma FPGA HARPv2 di Intel è possibile suddividere i layer di una rete convoluzionale su elementi di elaborazione separati per massimizzare la produttività. Sugli FPGA sono possibili anche pattern paralleli unici come le valutazioni di “ensemble” ad albero. Pipeline profonde con buffering e flusso di dati ottimizzati possono essere personalizzate in base alla struttura e ai tipi di dati di ogni modello. Questo livello di parallelismo e pipeline su misura non è fattibile sulle GPU.

10.3.2.1.3 Memoria On-Chip a Bassa Latenza

Grandi quantità di memoria on-chip ad alta larghezza di banda consentono l’archiviazione localizzata per pesi e attivazioni. Ad esempio, gli FPGA Xilinx Versal contengono 32 MB di blocchi RAM a bassa latenza e interfacce DDR4 a doppio canale per la memoria esterna. Avvicinare fisicamente la memoria alle unità di elaborazione riduce la latenza di accesso. Ciò fornisce significativi vantaggi di velocità rispetto alle GPU che attraversano PCIe (Peripheral Component Interconnect Express) o altri bus di sistema per raggiungere la memoria GDDR6 off-chip.

10.3.2.1.4 Supporto Nativo per Bassa Precisione

Un vantaggio fondamentale degli FPGA è la capacità di implementare in modo nativo qualsiasi larghezza di bit per unità aritmetiche, come INT4 o bfloat16, utilizzate nei modelli ML quantizzati. Ad esempio, Intel Stratix 10 NX FPGA ha core INT8 dedicati che possono raggiungere fino a 143 INT8 TOPS (Tera Operations Per Second) a ~1 TOPS/W (Tera Operations Per Second per Watt). TOPS è una misura di prestazioni simile a FLOPS, ma

mentre FLOPS misura calcoli in virgola mobile, TOPS misura il numero di operazioni intere che un sistema può eseguire al secondo. Le larghezze di bit inferiori, come INT8 o INT4, aumentano la densità aritmetica e le prestazioni. Gli FPGA possono persino supportare la sintonizzazione a precisione mista o dinamica in fase di esecuzione.

10.3.2.2 Svantaggi

10.3.2.2.1 Throughput di Picco Inferiore Rispetto agli ASIC

Gli FPGA non possono eguagliare i numeri di throughput grezzi degli ASIC, personalizzati per un modello e una precisione specifici. I sovraccarichi del “fabric” riconfigurabile rispetto all’hardware a funzione fissa comportano prestazioni di picco inferiori. Ad esempio, i pod TPU v5e consentono di connettere fino a 256 chip con oltre 100 PetaOps (Peta Operations Per Second) di prestazioni INT8, mentre gli FPGA possono offrire fino a 143 INT8 TOPS o 286 INT4 TOPS come sull’FPGA Intel Stratix 10 NX; PetaOps rappresenta quadrilioni di operazioni al secondo, mentre TOPS misura trilioni, evidenziando la capacità di elaborazione molto maggiore dei pod TPU rispetto agli FPGA.
Questo perché gli FPGA comprendono blocchi di base: blocchi logici configurabili, blocchi RAM e interconnessioni. Vengono forniti con una quantità stabilita di queste risorse. Per programmare gli FPGA, gli ingegneri scrivono codice HDL e lo compilano in flussi di bit che riorganizzano il “fabric”, che ha sovraccarichi intrinseci rispetto a un ASIC appositamente progettato per un calcolo.

10.3.2.2.2 Complessità di Programmazione

Per ottimizzare le prestazioni FPGA, gli ingegneri devono programmare le architetture in linguaggi di descrizione hardware di basso livello come Verilog o VHDL. Ciò richiede competenza nella progettazione hardware e cicli di sviluppo più lunghi rispetto a framework software di livello superiore come TensorFlow. Massimizzare l’utilizzo può essere difficile nonostante i progressi nella sintesi di alto livello da C/C++.

10.3.2.2.3 Sovraccarichi di Riconfigurazione

La modifica delle configurazioni FPGA richiede il ricaricamento di un nuovo flusso di bit, che ha costi di latenza e dimensioni di archiviazione considerevoli. Ad esempio, la riconfigurazione parziale su FPGA Xilinx può richiedere centinaia di millisecondi. Questo rende impossibile lo scambio dinamico di architetture in tempo reale. L’archiviazione del flusso di bit consuma anche memoria on-chip.

10.3.2.2.4 Guadagni in diminuzione sui nodi avanzati

Sebbene i nodi di processo più piccoli siano molto vantaggiosi per gli ASIC, offrono meno vantaggi per gli FPGA. A 7 nm e al di sotto, effetti come variazione di processo, vincoli termici e invecchiamento hanno un impatto sproporzionato sulle prestazioni degli FPGA. Anche le spese generali della struttura configurabile riducono i guadagni rispetto agli ASIC a funzione fissa.

10.3.3 Digital Signal Processor (DSP)

Il primo core di elaborazione del segnale digitale è stato costruito nel 1948 da Texas Instruments (The Evolution of Audio DSPs). Tradizionalmente, i DSP avrebbero avuto una logica per accedere direttamente ai dati digitali/audio nella memoria, eseguire un’operazione aritmetica (moltiplica-addiziona-accumula-MAC era una delle operazioni più comuni) e quindi scrivere il risultato nella memoria. Il DSP avrebbe incluso componenti analogici specializzati per recuperare i dati digitali/audio.
Una volta entrati nell’era degli smartphone, i DSP hanno iniziato a comprendere attività più sofisticate. Richiedevano Bluetooth, Wi-Fi e connettività cellulare. Anche i media sono diventati molto più complessi. Oggi, è raro avere chip interi dedicati solo al DSP, ma un

System on Chip includerebbe DSP e CPU per uso generico. Ad esempio, l'Hexagon Digital Signal Processor di Qualcomm afferma di essere un "processore di livello mondiale con funzionalità sia CPU che DSP per supportare le esigenze di elaborazione profondamente integrate della piattaforma mobile per funzioni sia multimediali che modem". Google Tensors, il chip nei telefoni Google Pixel, include anche CPU e motori DSP specializzati.

10.3.3.1 Vantaggi

I DSP offrono vantaggi architettonici in termini di throughput della matematica vettoriale, accesso alla memoria a bassa latenza, efficienza energetica e supporto per diversi tipi di dati, rendendoli adatti all'accelerazione ML embedded.

10.3.3.1.1 Architettura Ottimizzata per la Matematica Vettoriale

I DSP contengono percorsi dati specializzati, file di registro e istruzioni ottimizzati specificamente per le operazioni di matematica vettoriale comunemente utilizzate nei modelli di apprendimento automatico. Ciò include motori di prodotto scalare, unità MAC e funzionalità SIMD su misura per calcoli vettoriali/matriciali. Ad esempio, il DSP CEVA-XM6 ("Ceva SensPro fonde AI e Vector DSP") ha unità vettoriali a 512 bit per accelerare le convoluzioni. Questa efficienza sui carichi di lavoro di matematica vettoriale va ben oltre le CPU generiche.

10.3.3.1.2 Memoria On-Chip a Bassa Latenza

I DSP integrano grandi quantità di memoria SRAM veloce su chip per conservare i dati localmente per l'elaborazione. Avvicinare fisicamente la memoria alle unità di calcolo riduce la latenza di accesso. Ad esempio, il DSP SHARC+ di Analog contiene 10 MB di SRAM su chip. Questa memoria locale ad alta larghezza di banda offre vantaggi di velocità per le applicazioni in tempo reale.

10.3.3.1.3 Efficienza Energetica

I DSP sono progettati per fornire elevate prestazioni per watt su carichi di lavoro di segnali digitali. Percorsi dati efficienti, parallelismo e architetture di memoria consentono trilioni di operazioni matematiche al secondo entro budget di potenza mobili ridotti. Ad esempio, l'Hexagon DSP di Qualcomm può fornire 4 TOPS consumando un numero minimo di watt.

10.3.3.1.4 Supporto per Matematica a Virgola Mobile e Intera

A differenza delle GPU che eccellono in precisione singola o dimezzata, i DSP possono supportare nativamente tipi di dati a virgola mobile e intera a 8/16 bit utilizzati nei modelli ML. Alcuni DSP supportano l'accelerazione del prodotto scalare a precisione INT8 per reti neurali quantizzate.

10.3.3.2 Svantaggi

I DSP fanno compromessi architettonici che limitano il throughput di picco, la precisione e la capacità del modello rispetto ad altri acceleratori AI. Tuttavia, i loro vantaggi in termini di efficienza energetica e matematica intera li rendono una valida opzione di edge computing. Quindi, mentre i DSP offrono alcuni vantaggi rispetto alle CPU, presentano anche delle limitazioni per i carichi di lavoro di apprendimento automatico:

10.3.3.2.1 Throughput di Picco Inferiore Rispetto ad ASIC/GPU

I DSP non possono eguagliare il throughput computazionale grezzo delle GPU o degli ASIC personalizzati progettati specificamente per l'apprendimento automatico. Ad esempio, l'ASIC Cloud AI 100 di Qualcomm fornisce 480 TOPS su INT8, mentre il loro DSP Hexagon fornisce 10 TOPS. I DSP non hanno il massiccio parallelismo delle unità GPU SM.

10.3.3.2.2 Prestazioni a Doppia Precisione più Lente

La maggior parte dei DSP deve essere ottimizzata per la virgola mobile di precisione più elevata necessaria in alcuni modelli ML. I loro motori di prodotto scalare si concentrano su INT8/16 e FP32, che forniscono una migliore efficienza energetica. Tuttavia, la produttività in virgola mobile a 64 bit è molto più bassa, il che può limitare l'utilizzo nei modelli che richiedono un'elevata precisione.

10.3.3.2.3 Capacità del Modello Limitata

La limitata memoria on-chip dei DSP limita le dimensioni del modello che possono eseguire. Grandi modelli di deep learning con centinaia di megabyte di parametri supererebbero la capacità delle SRAM on-chip. I DSP sono più adatti per modelli di piccole e medie dimensioni destinati a dispositivi edge.

10.3.3.2.4 Complessità di Programmazione

La programmazione efficiente delle architetture DSP richiede competenza nella programmazione parallela e nell'ottimizzazione dei pattern di accesso ai dati. Le loro microarchitetture specializzate hanno una curva di apprendimento più ripida rispetto ai framework software di alto livello, rendendo lo sviluppo più complesso.

10.3.4 Graphics Processing Unit (GPU)

Il termine "graphics processing unit" [unità di elaborazione grafica] esiste almeno dagli anni '80. C'è sempre stata una richiesta di hardware grafico nelle console per videogiochi (elevata richiesta, doveva avere un costo relativamente basso) e nelle simulazioni scientifiche (richiesta inferiore, ma risoluzione più alta, poteva avere un prezzo elevato).
Il termine è stato reso popolare, tuttavia, nel 1999 quando NVIDIA ha lanciato la GeForce 256, mirando principalmente al settore di mercato dei giochi per PC (E. Lindholm et al. 2008). Man mano che i giochi per PC diventavano più sofisticati, le GPU NVIDIA diventavano più programmabili. Presto, gli utenti si resero conto che potevano sfruttare questa programmabilità, eseguire vari carichi di lavoro non correlati alla grafica sulle GPU e trarre vantaggio dall'architettura sottostante. E così, alla fine degli anni 2000, le GPU divennero unità di elaborazione grafica per uso generale o GP-GPU.
In seguito a questo cambiamento, altri importanti attori come Intel con la sua Arc Graphics e AMD con la sua serie Radeon RX hanno anche evoluto le loro GPU per supportare una gamma più ampia di applicazioni oltre al rendering grafico tradizionale. Questa espansione delle capacità delle GPU ha aperto nuove possibilità, in particolare nei campi che richiedono un'enorme potenza di calcolo.
Un esempio lampante di questo potenziale è la recente ricerca rivoluzionaria condotta da OpenAI (Brown et al. 2020) con GPT-3, un modello di linguaggio con 175 miliardi di parametri. L'addestramento di un modello così massiccio, che avrebbe richiesto mesi su CPU convenzionali, è stato completato in pochi giorni utilizzando potenti GPU, dimostrando l'impatto trasformativo delle GPU nell'accelerazione di complesse attività di apprendimento automatico.

10.3.4.1 Vantaggi

10.3.4.1.1 Elevata Capacità di Elaborazione

Il vantaggio principale delle GPU è la loro capacità di eseguire calcoli in virgola mobile paralleli massivi ottimizzati per la computer grafica e l'algebra lineare (Raina, Madhavan, e Ng 2009). Le GPU moderne come la A100 di Nvidia offrono fino a 19,5 teraflop di prestazioni FP32 con 6912 core CUDA e 40 GB di memoria grafica strettamente accoppiati a 1,6 TB/s di larghezza di banda della memoria grafica.

Questa capacità di elaborazione grezza deriva dall'architettura "Streaming Multiprocessor" (SM) altamente parallela, pensata per carichi di lavoro paralleli ai dati (Zhihao Jia, Zaharia, e Aiken 2019). Ogni SM contiene centinaia di core scalari ottimizzati per la matematica float32/64. Con migliaia di SM su un chip, le GPU sono appositamente progettate per la moltiplicazione di matrici e le operazioni vettoriali utilizzate in tutte le reti neurali.
Ad esempio, l'ultima GPU H100 di Nvidia fornisce 4000 TFLOP di FP8, 2000 TFLOP di FP16, 1000 TFLOP di TF32, 67 TFLOP di FP32 e 34 TFLOP di prestazioni di elaborazione FP64, che possono accelerare notevolmente l'addestramento di grandi batch su modelli come BERT, GPT-3 e altre architetture di trasformatori. Il parallelismo scalabile delle GPU è fondamentale per accelerare il deep learning computazionalmente intensivo.

10.3.4.1.2 Ecosistema Software Maturo

Nvidia fornisce ampie librerie di runtime come cuDNN e cuBLAS che sono altamente ottimizzate per primitive di deep learning. Framework come TensorFlow e PyTorch si integrano con queste librerie per abilitare l'accelerazione GPU senza programmazione diretta. Queste librerie sono basate su CUDA, la piattaforma di elaborazione parallela e il modello di programmazione di Nvidia.
CUDA (Compute Unified Device Architecture) è il framework sottostante che consente a queste librerie di alto livello di interagire con l'hardware della GPU. Fornisce agli sviluppatori un accesso di basso livello alle risorse della GPU, consentendo calcoli e ottimizzazioni personalizzate che sfruttano appieno le capacità di elaborazione parallela della GPU. Utilizzando CUDA, gli sviluppatori possono scrivere software che sfruttano l'architettura della GPU per attività di elaborazione ad alte prestazioni.
Questo ecosistema consente di sfruttare rapidamente le GPU ad alto livello tramite Python senza competenze di programmazione GPU. Flussi di lavoro e astrazioni noti forniscono una comoda rampa di accesso per scalare gli esperimenti di deep learning. La maturità del software integra i vantaggi della produttività.

10.3.4.1.3 Ampia Disponibilità

Le economie di scala dell'elaborazione grafica rendono le GPU ampiamente accessibili nei data center, nelle piattaforme cloud come AWS e GCP e nelle workstation desktop. La loro disponibilità negli ambienti di ricerca ha fornito una comoda piattaforma di sperimentazione e innovazione nel ML. Ad esempio, quasi tutti i risultati di deep learning all'avanguardia hanno coinvolto l'accelerazione GPU per merito di questa ubiquità. L'ampio accesso integra la maturità del software per rendere le GPU l'acceleratore ML standard.

10.3.4.1.4 Architettura Programmabile

Sebbene non siano flessibili come gli FPGA, le GPU offrono programmabilità tramite linguaggi CUDA e shader per personalizzare i calcoli. Gli sviluppatori possono ottimizzare i pattern di accesso ai dati, creare nuove operazioni e regolare le precisioni per modelli e algoritmi in evoluzione.

10.3.4.2 Svantaggi

Sebbene le GPU siano diventate l'acceleratore standard per il deep learning, la loro architettura presenta alcuni svantaggi importanti.

10.3.4.2.1 Meno Efficienti degli ASIC Custom

L'affermazione "Le GPU sono meno efficienti degli ASIC" potrebbe scatenare un acceso dibattito nel campo ML/AI e far esplodere questo libro.
In genere, le GPU sono percepite come meno efficienti degli ASIC perché questi ultimi sono realizzati su misura per attività specifiche e quindi possono funzionare in modo più efficiente nativamente. Con la loro architettura generica, le GPU sono intrinsecamente più versatili e programmabili, soddisfacendo un ampio spettro di attività computazionali oltre a ML/AI.

Tuttavia, le GPU moderne si sono evolute per includere un supporto hardware specializzato per operazioni AI essenziali, come la moltiplicazione di matrici generalizzata (GEMM) e altre operazioni di matrice, supporto nativo per la quantizzazione e supporto nativo per la potatura, che sono fondamentali per l'esecuzione efficace dei modelli ML. Questi miglioramenti hanno notevolmente migliorato l'efficienza delle GPU per le attività AI al punto che possono rivaleggiare con le prestazioni degli ASIC per determinate applicazioni.
Di conseguenza, le GPU contemporanee sono convergenti, incorporando capacità specializzate simili ad ASIC all'interno di un framework di elaborazione flessibile e di uso generale. Questa adattabilità ha offuscato i confini tra i due tipi di hardware. Le GPU offrono un forte equilibrio tra specializzazione e programmabilità che si adatta bene alle esigenze dinamiche della ricerca e sviluppo ML/AI.

10.3.4.2.2 Elevate Esigenze di Larghezza di Banda di Memoria

L'architettura massicciamente parallela richiede un'enorme larghezza di banda di memoria per alimentare migliaia di core. Ad esempio, la GPU Nvidia A100 richiede 1.6 TB/sec per saturare completamente il suo computer. Le GPU si affidano ad ampi bus di memoria a 384 bit per RAM GDDR6 ad alta larghezza di banda, ma anche la GDDR6 più veloce raggiunge il massimo a circa 1 TB/sec. Questa dipendenza dalla DRAM esterna comporta latenza e sovraccarico di potenza.

10.3.4.2.3 Complessità di Programmazione

Sebbene strumenti come CUDA siano utili, la mappatura e il partizionamento ottimali dei carichi di lavoro ML nell'architettura GPU massivamente parallela rimangono una sfida, il raggiungimento di un utilizzo elevato e della località della memoria richiede una messa a punto di basso livello (Zhe Jia et al. 2018). Astrazioni come TensorFlow possono tralasciare le prestazioni.

10.3.4.2.4 Memoria On-Chip Limitata

Le GPU hanno cache di memoria on-chip relativamente piccole rispetto ai grandi requisiti di working set dei modelli ML durante l'addestramento. Si basano su un accesso ad alta larghezza di banda alla DRAM esterna, che gli ASIC riducono al minimo con una grande SRAM on-chip.

10.3.4.2.5 Architettura Fissa

A differenza degli FPGA, l'architettura fondamentale della GPU non può essere modificata dopo la produzione. Questo vincolo limita l'adattamento a nuovi carichi di lavoro o layer ML. Il confine CPU-GPU crea anche overhead di spostamento dei dati.

10.3.5 Central Processing Unit (CPU)

Il termine CPU ha una lunga storia che risale al 1955 (Weik 1955) mentre la prima CPU a microprocessore, l'Intel 4004, è stata inventata nel 1971 (Chi ha inventato il microprocessore?). I compilatori traducono linguaggi di programmazione di alto livello come Python, Java o C per assemblare istruzioni (x86, ARM, RISC-V, ecc.) che le CPU devono elaborare. Il set di istruzioni che una CPU comprende è chiamato "instruction set architecture" (ISA), che definisce i comandi che il processore può eseguire direttamente. Deve essere concordato sia dall'hardware che dal software ci gira sopra.
Una panoramica degli sviluppi significativi nelle CPU:

- **Era del Single-core (anni '50-2000):** Questa era è nota per i miglioramenti microarchitettonici aggressivi. Tecniche come l'esecuzione speculativa (esecuzione di un'istruzione prima che quella precedente fosse finita), "out-of-order execution" [esecuzione fuori ordine] (riordinamento delle istruzioni per renderle più efficaci) e "wider issue widths" [larghezze di emissione più ampie] (esecuzione di più istruzioni

contemporaneamente) sono state implementate per aumentare la produttività delle istruzioni. Anche il termine "System on Chip" ha avuto origine in questa era, poiché diversi componenti analogici (componenti progettati con transistor) e componenti digitali (componenti progettati con linguaggi di descrizione hardware mappati su transistor) sono stati inseriti sulla stessa piattaforma per realizzare un'attività.

- **Era Multicore (anni 2000):** Guidata dalla diminuzione della legge di Moore, questa è caratterizzata dall'aumento del numero di core all'interno di una CPU. Ora, le attività possono essere suddivise su più core diversi, ognuno con il proprio percorso dati e unità di controllo. Molti dei problemi di quest'epoca riguardavano come condividere determinate risorse, quali risorse condividere e come mantenere coerenza e consistenza in tutti i core.
- **Un Mare di acceleratori (anni 2010):** Ancora una volta, spinta dalla diminuzione della legge di Moore, quest'epoca è caratterizzata dal delegare le attività più complicate su acceleratori (widget) collegati al datapath principale nelle CPU. È comune vedere acceleratori dedicati a vari carichi di lavoro di intelligenza artificiale, nonché elaborazione di immagini/digitali e crittografia. In queste progettazioni, le CPU sono spesso descritte più come giudici, che decidono quali attività devono essere elaborate piuttosto che eseguire l'elaborazione stessa. Qualsiasi attività potrebbe comunque essere eseguita sulla CPU anziché sugli acceleratori, ma la CPU sarebbe generalmente più lenta. Tuttavia, il costo di progettazione e programmazione dell'acceleratore è diventato un ostacolo non banale che ha suscitato interesse per le librerie specifiche per la progettazione (DSL).
- **Presenza nei data center:** Sebbene sentiamo spesso dire che le GPU dominano il mercato dei data center, le CPU sono comunque adatte per attività che non possiedono intrinsecamente un elevato grado di parallelismo. Le CPU spesso gestiscono attività seriali e di piccole dimensioni e coordinano il data center.
- **Sull'edge:** Dati i vincoli più rigidi sulle risorse sull'edge, le CPU edge spesso implementano solo un sottoinsieme delle tecniche sviluppate nell'era single-core perché queste ottimizzazioni tendono a essere pesanti in termini di consumo di energia e area. Le CPU edge mantengono comunque un datapath relativamente semplice con capacità di memoria limitate.

Tradizionalmente, le CPU sono state sinonimo di elaborazione generica, un termine che è cambiato anche perché il carico di lavoro "medio" che un consumatore esegue cambia nel tempo. Ad esempio, i componenti in virgola mobile erano un tempo considerati riservati alla "elaborazione scientifica", di solito venivano implementati come un coprocessore (un componente modulare che funzionava con il datapath) e raramente distribuiti ai consumatori medi. Confrontate questo atteggiamento con quello odierno, in cui le FPU sono integrate in ogni datapath.

10.3.5.1 Vantaggi

Sebbene la produttività in sé sia limitata, le CPU per uso generico offrono vantaggi pratici di accelerazione AI.

10.3.5.1.1 Programmabilità Generale

Le CPU supportano carichi di lavoro diversi oltre al ML, offrendo una programmabilità flessibile per uso generico. Questa versatilità deriva dai loro set di istruzioni standardizzati e dagli ecosistemi di compilatori maturi, che consentono di eseguire qualsiasi applicazione, dai database e server Web alle pipeline analitiche (Hennessy e Patterson 2019).
Questo evita la necessità di acceleratori ML dedicati e consente di sfruttare l'infrastruttura basata su CPU esistenti per la distribuzione ML di base. Ad esempio, i server X86 di fornitori

come Intel e AMD possono eseguire framework ML comuni utilizzando pacchetti Python e TensorFlow insieme ad altri carichi di lavoro aziendali.

10.3.5.1.2 Ecosistema Software Maturo

Per decenni, librerie matematiche altamente ottimizzate come BLAS, LAPACK e FFTW hanno sfruttato istruzioni vettorializzate e multithreading su CPU (Dongarra 2009). I principali framework ML come PyTorch, TensorFlow e SciKit-Learn sono progettati per integrarsi perfettamente con questi kernel matematici di CPU.
I fornitori di hardware come Intel e AMD forniscono anche librerie di basso livello per ottimizzare completamente le prestazioni per primitive di deep learning (accelerazione dell'inferenza AI su CPU). Questo ecosistema software robusto e maturo consente di distribuire rapidamente ML su infrastrutture di CPU esistenti.

10.3.5.1.3 Ampia Disponibilità

Le economie di scala della produzione di CPU, guidate dalla domanda in molti mercati come PC, server e dispositivi mobili, le rendono disponibili ovunque. Le CPU Intel, ad esempio, hanno alimentato la maggior parte dei server per decenni (Ranganathan 2011). Questa ampia disponibilità nei data center riduce i costi hardware per l'implementazione di ML di base. Anche i piccoli dispositivi embedded in genere integrano una certa CPU, consentendo l'inferenza edge. L'ubiquità riduce la necessità di acquistare acceleratori ML specializzati in molte situazioni.

10.3.5.1.4 Basso Consumo per L'inferenza

Ottimizzazioni come ARM Neon e le estensioni vettoriali Intel AVX forniscono un throughput di numeri interi e in virgola mobile a basso consumo ottimizzato per carichi di lavoro "a raffica" come l'inferenza (Ignatov et al. 2018). Sebbene più lenta delle GPU, l'inferenza CPU può essere implementata in ambienti con vincoli energetici. Ad esempio, le CPU Cortex-M di ARM ora offrono oltre 1 TOPS di prestazioni INT8 sotto 1 W, consentendo l'individuazione di parole chiave e applicazioni di visione su dispositivi edge (ARM).

10.3.5.2 Svantaggi

Pur offrendo alcuni vantaggi, le CPU per uso generico presentano anche delle limitazioni per i carichi di lavoro AI.

10.3.5.2.1 Throughput Inferiore Rispetto agli Acceleratori

Le CPU non dispongono delle architetture specializzate per l'elaborazione parallela massiva che GPU e altri acceleratori forniscono. Il loro design per uso generico riduce il throughput computazionale per le operazioni matematiche altamente parallelizzabili comuni nei modelli ML (N. P. Jouppi et al. 2017a).

10.3.5.2.2 Non Ottimizzato per il Parallelismo dei Dati

Le architetture delle CPU non sono specificamente ottimizzate per i carichi di lavoro paralleli dei dati inerenti all'AI (Sze et al. 2017). Assegnano un'area di silicio sostanziale alla decodifica delle istruzioni, all'esecuzione speculativa, alla memorizzazione nella cache e al controllo del flusso che fornisce pochi vantaggi per le operazioni su array utilizzate nelle reti neurali (accelerazione dell'inferenza AI sulle CPU). Tuttavia, le CPU moderne sono dotate di istruzioni vettoriali come AVX-512 specificamente per accelerare determinate operazioni chiave come la moltiplicazione matriciale.
I multiprocessori di streaming GPU, ad esempio, dedicano la maggior parte dei transistor alle unità a virgola mobile anziché alla logica di predizione di diramazione complessa. Questa specializzazione consente un utilizzo molto più elevato per la matematica ML.

10.3.5.2.3 Maggiore Latenza della Memoria

Le CPU soffrono di una latenza maggiore nell'accesso alla memoria principale rispetto alle GPU e ad altri acceleratori (DDR). Tecniche come il tiling e il caching possono aiutare, ma la separazione fisica dalla RAM off-chip crea colli di bottiglia nei carichi di lavoro ML ad alta intensità di dati. Ciò sottolinea la necessità di architetture di memoria specializzate nell'hardware ML.

10.3.5.2.4 Inefficienza Energetica in Caso di Carichi di Lavoro Pesanti

Sebbene sia adatto per l'inferenza intermittente, il mantenimento di una produttività quasi di picco per l'addestramento comporta un consumo energetico inefficiente sulle CPU, in particolare sulle CPU mobili (Ignatov et al. 2018). Gli acceleratori ottimizzano esplicitamente il flusso di dati, la memoria e il calcolo per carichi di lavoro ML sostenuti. Le CPU sono inefficienti dal punto di vista energetico per l'addestramento di modelli di grandi dimensioni.

10.3.6 Confronto

Tabella 10.2 confronta i diversi tipi di funzionalità hardware.

Tabella 10.2: Confronto di diversi acceleratori hardware per carichi di lavoro AI.

Acceleratore	Descrizione	Principali vantaggi	Principali svantaggi
ASIC	IC personalizzati progettati per carichi di lavoro target come l'inferenza AI	• Massimizza le prestazioni/watt. • Ottimizzato per le operazioni tensoriali • Memoria on-chip a bassa latenza	• L'architettura fissa manca di flessibilità • Elevato costo NRE • Lunghi cicli di progettazione
FPGA	Fabric riconfigurabile con logica programmabile e routing	• Architettura flessibile • Accesso alla memoria a bassa latenza	• Prestazioni/watt inferiori rispetto agli ASIC • Programmazione complessa
GPU	Originariamente per la grafica, ora utilizzate per l'accelerazione della rete neurale	• Elevata produttività • Scalabilità parallela • Ecosistema software con CUDA	• Non efficienti dal punto di vista energetico come gli ASIC • Richiede un'elevata larghezza di banda della memoria
CPU	Processori per uso generico	• Programmabilità • Disponibilità ubiqua	• Prestazioni inferiori per carichi di lavoro AI

In generale, le CPU forniscono una baseline prontamente disponibile, le GPU offrono un'accelerazione ampiamente accessibile, gli FPGA offrono programmabilità e gli ASIC massimizzano l'efficienza per funzioni fisse. La scelta ottimale dipende dalla scala, dal costo, dalla flessibilità e da altri requisiti dell'applicazione target.

Sebbene inizialmente sviluppati per l'implementazione del data center, Google ha anche profuso notevoli sforzi nello sviluppo di TPU Edge. Questi TPU Edge mantengono l'ispirazione degli array sistolici [https://it.wikipedia.org/wiki/Array_sistolico], ma sono adattati alle risorse limitate accessibili all'edge.

10.4 Co-Progettazione Hardware-Software

La co-progettazione hardware-software si basa sul principio secondo cui i sistemi AI raggiungono prestazioni ed efficienza ottimali quando i componenti hardware e software

sono progettati in stretta integrazione. Ciò comporta un ciclo di progettazione iterativo e collaborativo in cui l'architettura hardware e gli algoritmi software vengono sviluppati e perfezionati contemporaneamente con un feedback continuo tra i team.
Ad esempio, un nuovo modello di rete neurale può essere prototipato su una piattaforma di accelerazione basata su FPGA per ottenere dati sulle prestazioni reali all'inizio del processo di progettazione. Questi risultati forniscono un feedback ai progettisti hardware su potenziali ottimizzazioni e agli sviluppatori software su perfezionamenti del modello o framework per sfruttare meglio le capacità hardware. Questo livello di sinergia è difficile da raggiungere con la pratica comune di software sviluppato in modo indipendente per essere distribuito su hardware fisso.
La progettazione congiunta è fondamentale per i sistemi di intelligenza artificiale embedded che affrontano notevoli vincoli di risorse come budget di potenza ridotti, memoria e capacità di elaborazione limitate e requisiti di latenza in tempo reale. La stretta integrazione tra sviluppatori di algoritmi e architetti hardware aiuta a sbloccare le ottimizzazioni in tutto lo stack per soddisfare queste restrizioni. Le tecniche di abilitazione includono miglioramenti algoritmici come la ricerca e il pruning [potatura] dell'architettura neurale e progressi hardware come flussi di dati specializzati e gerarchie di memoria.
Riunendo la progettazione hardware e software, anziché svilupparli separatamente, è possibile realizzare ottimizzazioni olistiche che massimizzano prestazioni ed efficienza. Le sezioni successive forniscono maggiori dettagli su specifici approcci di progettazione congiunta.

10.4.1 La Necessità della Progettazione Congiunta

Diversi fattori chiave rendono essenziale un approccio di progettazione congiunta hardware-software collaborativo per la creazione di sistemi di intelligenza artificiale efficienti.

10.4.1.1 Aumento delle Dimensioni e della Complessità del Modello

I modelli di intelligenza artificiale all'avanguardia sono cresciuti rapidamente in termini di dimensioni, abilitati dai progressi nella progettazione dell'architettura neurale e dalla disponibilità di grandi set di dati. Ad esempio, il modello linguistico GPT-3 contiene 175 miliardi di parametri (Brown et al. 2020), che richiedono enormi risorse di calcolo per l'addestramento. Questa esplosione nella complessità del modello richiede una progettazione congiunta per sviluppare hardware e algoritmi efficienti in tandem. Tecniche come la compressione del modello (Y. Cheng et al. 2018) e la quantizzazione devono essere co-ottimizzate con l'architettura hardware.

10.4.1.2 Vincoli della Distribuzione Embedded

L'implementazione di applicazioni AI su dispositivi edge come telefoni cellulari o elettrodomestici intelligenti introduce vincoli significativi su energia, memoria e area di silicio (Sze et al. 2017). Abilitare l'inferenza in tempo reale con queste restrizioni richiede la co-esplorazione di ottimizzazioni hardware come flussi di dati specializzati e compressione con progettazione efficiente di reti neurali e tecniche di potatura. La co-progettazione massimizza le prestazioni entro rigidi vincoli di distribuzione.

10.4.1.3 Rapida Evoluzione degli Algoritmi AI

L'intelligenza artificiale si sta evolvendo rapidamente, con nuove architetture di modelli, metodologie di training e framework software che emergono costantemente. Ad esempio, i Transformers sono diventati di recente molto popolari per l'NLP (Young et al. 2018). Per tenere il passo con queste innovazioni algoritmiche è necessaria una progettazione congiunta hardware-software per adattare le piattaforme ed evitare rapidamente il debito tecnico accumulato.

10.4.1.4 Interazioni Complesse Hardware-Software

Molte interazioni e compromessi sottili tra scelte architettoniche hardware e ottimizzazioni software hanno un impatto significativo sull'efficienza complessiva. Ad esempio, tecniche come il partizionamento tensoriale e il batching influenzano il parallelismo e i pattern di accesso ai dati influenzano l'utilizzo della memoria. La progettazione congiunta fornisce una prospettiva multilivello per svelare queste dipendenze.

10.4.1.5 Necessità di Specializzazione

I carichi di lavoro dell'intelligenza artificiale traggono vantaggio da operazioni specializzate come matematica a bassa precisione e gerarchie di memoria personalizzate. Ciò motiva l'incorporazione di hardware personalizzato su misura per algoritmi di reti neurali piuttosto che affidarsi esclusivamente a software flessibile in esecuzione su hardware generico (Sze et al. 2017). Tuttavia, lo stack software deve mirare esplicitamente alle operazioni hardware personalizzate per realizzare i vantaggi.

10.4.1.6 Richiesta di Maggiore Efficienza

Con la crescente complessità del modello, si verificano rendimenti decrescenti e spese generali derivanti dall'ottimizzazione del solo hardware o software in isolamento (Putnam et al. 2014). Si presentano inevitabili compromessi che richiedono un'ottimizzazione globale su più livelli. La progettazione congiunta di hardware e software fornisce grandi guadagni di efficienza composti.

10.4.2 Principi di Progettazione Congiunta Hardware-Software

L'architettura hardware e lo stack software devono essere strettamente integrati e co-ottimizzati per creare sistemi di intelligenza artificiale efficienti e ad alte prestazioni. Nessuno dei due può essere progettato in isolamento; massimizzare le loro sinergie richiede un approccio olistico noto come progettazione congiunta hardware-software.
L'obiettivo principale è adattare le capacità hardware in modo che corrispondano agli algoritmi e ai carichi di lavoro eseguiti dal software. Ciò richiede un ciclo di feedback tra architetti hardware e sviluppatori software per convergere su soluzioni ottimizzate. Diverse tecniche consentono un'efficace co-progettazione:

10.4.2.1 Ottimizzazione Software Consapevole dell'Hardware

Lo stack software può essere ottimizzato per sfruttare meglio le capacità hardware:

- **Parallelismo:** Parallelizzare i calcoli matriciali come convoluzione o layer di attenzione per massimizzare la produttività sui motori vettoriali.
- **Ottimizzazione della Memoria:** Ottimizzare i layout dei dati per migliorare la località della cache in base alla profilazione hardware. Ciò massimizza il riutilizzo e riduce al minimo l'accesso DRAM costoso.
- **Compressione:** Utilizzare la sparsity [diradazione] nei modelli per ridurre lo spazio di archiviazione e risparmiare sui calcoli tramite operazioni di zero-skipping.
- **Operazioni Personalizzate:** Incorporare operazioni specializzate come INT4 a bassa precisione o bfloat16 nei modelli per sfruttare al meglio il supporto hardware dedicato.
- **Mappatura del Flusso di Dati:** Mappare esplicitamente le fasi del modello alle unità di calcolo per ottimizzare lo spostamento dei dati sull'hardware.

10.4.2.2 Specializzazione Hardware Algorithm-Driven

L'hardware può essere adattato alle caratteristiche degli algoritmi ML:

- **Tipi di Dati Personalizzati:** Supportare INT8/4 o bfloat16 a bassa precisione nell'hardware per una maggiore densità aritmetica.

- **Memoria su Chip:** Aumentare la larghezza di banda SRAM e ridurre la latenza di accesso per adattarla ai pattern di accesso alla memoria del modello.
- **Operazioni Specifiche del Dominio:** Aggiungere unità hardware per funzioni ML chiave come FFT o moltiplicazione di matrici per ridurre latenza ed energia.
- **Profilazione del Modello:** Utilizzare la simulazione e la profilazione del modello per identificare hotspot computazionali e ottimizzare l'hardware.

La chiave è il feedback collaborativo: le informazioni dalla profilazione dell'hardware guidano le ottimizzazioni del software, mentre i progressi algoritmici informano la specializzazione dell'hardware. Questo miglioramento reciproco fornisce guadagni di efficienza moltiplicativa rispetto agli sforzi isolati.

10.4.2.3 Co-esplorazione Algoritmo-Hardware

Una potente tecnica di co-progettazione prevede l'esplorazione congiunta di innovazioni nelle architetture di reti neurali e nella progettazione custom dell'hardware. A powerful co-design technique involves jointly exploring innovations in neural network architectures and custom hardware design. Ciò consente di trovare abbinamenti ideali su misura per i rispettivi punti di forza (Sze et al. 2017).

Ad esempio, il passaggio ad architetture mobili come MobileNets (Howard et al. 2017) è stato guidato dai vincoli dei dispositivi edge come dimensioni del modello e latenza. La quantizzazione (Jacob et al. 2018) e le tecniche di pruning [potatura] (Gale, Elsen, e Hooker 2019) che hanno reso questi modelli efficienti sono diventate possibili grazie ad acceleratori hardware con supporto nativo per interi a bassa precisione e supporto per potatura (Mishra et al. 2021).

I modelli basati sull'attenzione hanno prosperato su GPU e ASIC massivamente paralleli, dove il loro calcolo si mappa bene nello spazialmente, al contrario delle architetture RNN, che si basano sull'elaborazione sequenziale. La co-evoluzione di algoritmi e hardware ha evidenziato nuove capacità.

Una co-esplorazione efficace richiede una stretta collaborazione tra ricercatori di algoritmi e architetti hardware. La prototipazione rapida su FPGA (Chen Zhang et al. 2015) o simulatori di intelligenza artificiale specializzati consente una rapida valutazione di diverse coppie di architetture di modelli e progetti hardware pre-silicio.

Ad esempio, l'architettura TPU di Google si è evoluta con ottimizzazioni verso i modelli TensorFlow per massimizzare le prestazioni sulla classificazione delle immagini. Questo stretto ciclo di feedback ha prodotto modelli su misura per la TPU che sarebbero stati improbabili in isolamento.

Gli studi hanno mostrato guadagni di prestazioni ed efficienza da 2 a 5 volte superiori con la co-esplorazione algoritmo-hardware rispetto agli sforzi isolati di ottimizzazione di algoritmi o hardware (Suda et al. 2016). Parallelizzare lo sviluppo congiunto riduce anche i "time-to-deployment" [tempi di distribuzione].

Nel complesso, esplorare le strette interdipendenze tra innovazione del modello e progressi hardware crea opportunità che devono essere visibili quando vengono affrontate in sequenza. Questa progettazione sinergica congiunta produce soluzioni maggiori della somma delle loro parti.

10.4.3 Sfide

Sebbene la progettazione collaborativa possa migliorare l'efficienza, l'adattabilità e il time-to-market, presenta anche sfide ingegneristiche e organizzative.

10.4.3.1 Aumento dei Costi di Prototipazione

È richiesta una prototipazione più estesa per valutare diverse accoppiate hardware-software. La necessità di prototipi rapidi e iterativi su FPGA o emulatori aumenta il sovraccarico della

validazione. Ad esempio, Microsoft ha scoperto che erano necessari più prototipi per la progettazione collaborativa di un acceleratore AI rispetto alla progettazione sequenziale (Fowers et al. 2018).

10.4.3.2 Ostacoli Organizzativi e di Team

La progettazione collaborativa richiede uno stretto coordinamento tra gruppi hardware e software tradizionalmente scollegati. Ciò potrebbe causare problemi di comunicazione o priorità e pianificazioni non allineate. Anche la navigazione di diversi flussi di lavoro di progettazione è impegnativa. Potrebbe esistere una certa inerzia organizzativa nell'adottare pratiche integrate.

10.4.3.3 Complessità di Simulazione e Modellazione

Catturare interazioni sottili tra layer hardware e software per la simulazione e la modellazione congiunte aggiunge una complessità significativa. Le astrazioni complete "cross-layer" sono difficili da costruire quantitativamente prima dell'implementazione, rendendo più difficile quantificare in anticipo le ottimizzazioni olistiche.

10.4.3.4 Rischi di Eccessiva Specializzazione

Una progettazione congiunta rigorosa comporta il rischio di adattare eccessivamente le ottimizzazioni agli algoritmi correnti, sacrificando la generalità. Ad esempio, l'hardware ottimizzato esclusivamente per i modelli Transformer potrebbe avere prestazioni inferiori con le tecniche future. Mantenere la flessibilità richiede lungimiranza.

10.4.3.5 Problemi sui Cambiamenti

Gli ingegneri che hanno familiarità con le consolidate pratiche di progettazione hardware o software discrete potrebbero accettare solo flussi di lavoro collaborativi familiari. Nonostante i vantaggi a lungo termine, i progetti potrebbero incontrare attriti nella transizione alla progettazione congiunta.

10.5 Software per Hardware AI

Acceleratori hardware specializzati come GPU, TPU e FPGA sono essenziali per fornire applicazioni di intelligenza artificiale ad alte prestazioni. Tuttavia, è necessario un ampio stack software per sfruttare efficacemente queste piattaforme hardware, che coprano l'intero ciclo di vita di sviluppo e distribuzione. Framework e librerie costituiscono la spina dorsale dell'hardware AI, offrendo set di codice, algoritmi e funzioni pre-costruiti e robusti, specificamente ottimizzati per eseguire varie attività AI su hardware diversi. Sono progettati per semplificare le complessità dell'utilizzo dell'hardware da zero, che può richiedere molto tempo ed essere soggetto a errori. Il software svolge un ruolo importante:

- Fornendo astrazioni di programmazione e modelli come CUDA e OpenCL per mappare i calcoli sugli acceleratori.
- Integrando gli acceleratori in framework di deep learning popolari come TensorFlow e PyTorch.
- Ottimizzando l'intero stack hardware-software con compilatori e tool.
- Con piattaforme di simulazione per modellare insieme hardware e software.
- Con l'infrastruttura per gestire la distribuzione sugli acceleratori.

Questo vasto ecosistema software è importante quanto l'hardware nel fornire applicazioni AI performanti ed efficienti. Questa sezione fornisce una panoramica degli strumenti disponibili a ogni livello dello stack per consentire agli sviluppatori di creare ed eseguire sistemi AI basati sull'accelerazione hardware.

10.5.1 Modelli di Programmazione

I modelli di programmazione forniscono astrazioni per mappare calcoli e dati su acceleratori hardware eterogenei:

- **CUDA:** Modello di programmazione parallela di Nvidia per sfruttare le GPU utilizzando estensioni a linguaggi come C/C++. Consente di avviare kernel su core GPU (Luebke 2008).
- **OpenCL:** Standard aperto per scrivere programmi che spaziano tra CPU, GPU, FPGA e altri acceleratori. Specifica un framework di elaborazione eterogeneo (Munshi 2009).
- **OpenGL/WebGL:** Interfacce di programmazione grafica 3D in grado di mappare codice generico su core GPU (Segal e Akeley 1999).
- **Verilog/VHDL:** "Hardware description languages (HDL)" [Linguaggi di descrizione hardware] utilizzati per configurare FPGA come acceleratori AI specificando circuiti digitali (Gannot e Ligthart 1994).
- **TVM:** Un framework di compilazione che fornisce un frontend Python per ottimizzare e mappare modelli di deep learning su diversi backend hardware (T. Chen et al. 2018).

Le sfide principali includono l'espressione del parallelismo, la gestione della memoria tra dispositivi e l'abbinamento di algoritmi alle capacità hardware. Le astrazioni devono bilanciare la portabilità con la possibilità di personalizzazione hardware. I modelli di programmazione consentono agli sviluppatori di sfruttare gli acceleratori senza competenze hardware. Questi dettagli sono discussi nella sezione AI frameworks section.

Esercizio 10.1: Software per hardware AI - TVM

Abbiamo imparato che l'hardware AI sofisticato ha bisogno di un software speciale per fare magie. TVM è come un traduttore super intelligente, che trasforma il codice in istruzioni che gli acceleratori capiscono. In questo Colab, useremo TVM per creare un acceleratore finto chiamato VTA che esegue la moltiplicazione di matrici super velocemente. Pronti a vedere come il software alimenta l'hardware?

Open in Colab

10.5.2 Librerie e Runtime

Librerie e runtime specializzati forniscono astrazioni software per accedere e massimizzare l'utilizzo degli acceleratori AI:

- **Librerie Matematiche:** Implementazioni altamente ottimizzate di primitive di algebra lineare come GEMM, FFT, convoluzioni, ecc., su misura per l'hardware target. Nvidia cuBLAS, Intel MKL e librerie di elaborazione Arm sono esempi.
- **Integrazioni di Framework:** Librerie per accelerare framework di deep learning come TensorFlow, PyTorch e MXNet su hardware supportato. Ad esempio, cuDNN accelera le CNN sulle GPU Nvidia.
- **Runtime:** Software per gestire l'esecuzione dell'acceleratore, tra cui pianificazione, sincronizzazione, gestione della memoria e altre attività. Nvidia TensorRT è un ottimizzatore di inferenza e runtime.
- **Driver e firmware:** Software di basso livello per interfacciarsi con l'hardware, inizializzare i dispositivi e gestire l'esecuzione. Fornitori come Xilinx forniscono driver per le loro schede acceleratrici.

Ad esempio, gli integratori PyTorch utilizzano librerie cuDNN e cuBLAS per accelerare l'addestramento sulle GPU Nvidia. Il runtime TensorFlow XLA ottimizza e compila modelli per acceleratori come le TPU. I driver inizializzano i dispositivi e delegano le operazioni.

Le sfide includono il partizionamento e la pianificazione efficienti dei carichi di lavoro su dispositivi eterogenei come nodi multi-GPU. I runtime devono anche ridurre al minimo il sovraccarico dei trasferimenti di dati e della sincronizzazione.
Librerie, runtime e driver forniscono i mattoni ottimizzati che gli sviluppatori di deep learning possono sfruttare per le prestazioni dell'acceleratore senza competenze di programmazione hardware. La loro ottimizzazione è essenziale per le distribuzioni.

10.5.3 Ottimizzazione dei Compilatori

L'ottimizzazione dei compilatori è fondamentale per estrarre le massime prestazioni ed efficienza dagli acceleratori hardware per i carichi di lavoro AI. Applicano ottimizzazioni che spaziano tra modifiche algoritmiche, trasformazioni a livello di grafico e generazione di codice di basso livello.

- **Ottimizzazione degli Algoritmi:** Tecniche come quantizzazione, potatura e ricerca di architettura neurale per migliorare l'efficienza del modello e abbinare le capacità hardware.
- **Ottimizzazioni dei Grafi:** Ottimizzazioni a livello di grafo come fusione degli operatori, riscrittura e trasformazioni di layout per ottimizzare le prestazioni sull'hardware target.
- **Generazione di Codice:** Generazione di codice di basso livello ottimizzato per acceleratori da modelli e framework di alto livello.

Ad esempio, lo stack di compilatori "open" TVM applica la quantizzazione per un modello BERT che ha come target le GPU Arm. Fonde le operazioni di convoluzione puntuale e trasforma il layout dei pesi per ottimizzare l'accesso alla memoria. Infine, emette codice OpenGL ottimizzato per eseguire il carico di lavoro GPU.
Le ottimizzazioni chiave del compilatore includono la massimizzazione del parallelismo, il miglioramento della località e del riutilizzo dei dati, la riduzione al minimo dell'ingombro della memoria e lo sfruttamento delle operazioni hardware personalizzate. I compilatori creano e ottimizzano i carichi di lavoro di machine learning in modo olistico su componenti hardware come CPU, GPU e altri acceleratori.
Tuttavia, la mappatura efficiente di modelli complessi introduce sfide come il partizionamento efficiente dei carichi di lavoro su dispositivi eterogenei. I compilatori a livello di produzione richiedono anche molto tempo per la messa a punto su carichi di lavoro rappresentativi. Tuttavia, l'ottimizzazione dei compilatori è per sfruttare tutte le capacità degli acceleratori AI.

10.5.4 Simulazione e Modellazione

Il software di simulazione è importante nella progettazione congiunta hardware-software. Consente accoppiare la modellazione di architetture hardware e stack software proposti:

- **Simulazione Hardware:** Piattaforme come Gem5 consentono la simulazione dettagliata di componenti hardware come pipeline, cache, interconnessioni e gerarchie di memoria. Gli ingegneri possono modellare le modifiche hardware senza prototipazione fisica (Binkert et al. 2011).
- **Simulazione Software:** Stack di compilatori come TVM supportano la simulazione di carichi di lavoro di machine learning per stimare le prestazioni sulle architetture hardware target. Questo aiuta con le ottimizzazioni software.
- **Co-simulazione:** Piattaforme unificate come SCALE-Sim (Samajdar et al. 2018) integrano la simulazione hardware e software in un unico strumento. Ciò consente un'analisi "what-if" per quantificare gli impatti a livello di sistema delle ottimizzazioni cross-layer all'inizio del ciclo di progettazione.

Ad esempio, un progetto di acceleratore AI basato su FPGA potrebbe essere simulato utilizzando il linguaggio di descrizione hardware Verilog e sintetizzato in un modello Gem5. Verilog è adatto per descrivere la logica digitale e le interconnessioni dell'architettura dell'acceleratore. Verilog consente al progettista di specificare i datapath [percorsi dati], la logica di controllo, le memorie on-chip e altri componenti implementati nella struttura FPGA. Una volta completato il progetto Verilog, può essere sintetizzato in un modello che simula il comportamento dell'hardware, ad esempio utilizzando il simulatore Gem5. Gem5 è utile per questa attività perché consente la modellazione di sistemi completi, inclusi processori, cache, bus e acceleratori personalizzati. Gem5 supporta l'interfacciamento dei modelli Verilog dell'hardware alla simulazione, consentendo la modellazione unificata del sistema.
Il modello di acceleratore FPGA sintetizzato potrebbe quindi avere carichi di lavoro ML simulati utilizzando TVM compilato su di esso all'interno dell'ambiente Gem5 per una modellazione unificata. TVM consente la compilazione ottimizzata di modelli di ML su hardware eterogeneo come FPGA. L'esecuzione di carichi di lavoro compilati con TVM sull'acceleratore all'interno della simulazione Gem5 fornisce un modo integrato per convalidare e perfezionare la progettazione hardware, lo stack software e l'integrazione di sistema prima di realizzare fisicamente l'acceleratore su un FPGA reale.
Questo tipo di co-simulazione fornisce stime di metriche complessive come throughput, latenza e potenza per guidare la progettazione congiunta prima della costosa prototipazione fisica. Aiutano anche con le ottimizzazioni di partizionamento tra hardware e software per guidare i compromessi di progettazione.
Tuttavia, la precisione nella modellazione di interazioni sottili di basso livello tra componenti è limitata. Le simulazioni quantificate sono stime ma non possono sostituire completamente i prototipi fisici e i test. Tuttavia, la simulazione e la modellazione unificate forniscono preziose informazioni iniziali sulle opportunità di ottimizzazione a livello di sistema durante il processo di co-progettazione.

10.6 Benchmarking dell'Hardware AI

Il benchmarking è un processo critico che quantifica e confronta le prestazioni di varie piattaforme hardware progettate per accelerare le applicazioni di intelligenza artificiale. Guida le decisioni di acquisto, l'attenzione allo sviluppo e gli sforzi di ottimizzazione delle prestazioni per i produttori di hardware e gli sviluppatori di software.
Il capitolo sul benchmarking esplora questo argomento in modo molto dettagliato, spiegando perché è diventato una parte indispensabile del ciclo di sviluppo dell'hardware AI e come influisce sul più ampio panorama tecnologico. Qui, esamineremo brevemente i concetti principali, ma consigliamo di fare riferimento al capitolo per maggiori dettagli.
Suite di benchmarking come MLPerf, Fathom e AI Benchmark offrono una serie di test standardizzati utilizzabili su diverse piattaforme hardware. Queste suite misurano le prestazioni dell'acceleratore AI su varie reti neurali e attività di apprendimento automatico, dalla classificazione di immagini di base all'elaborazione complessa del linguaggio. Fornendo un terreno comune per il confronto, aiutano a garantire che le dichiarazioni sulle prestazioni siano coerenti e verificabili. Questi "tool" vengono applicati non solo per guidare lo sviluppo dell'hardware, ma anche per garantire che lo stack software sfrutti appieno il potenziale dell'architettura sottostante.

- **MLPerf:** Include un ampio set di benchmark che coprono sia l'addestramento (Mattson et al. 2020a) che l'inferenza (V. J. Reddi et al. 2020) per una gamma di attività di machine learning. Figura 11.5 illustra la diversità dei casi d'uso dell'IA trattati da MLPerf.
- **Fathom:** Si concentra sulle operazioni principali nei modelli di deep learning, enfatizzandone l'esecuzione su diverse architetture (Adolf et al. 2016).

- **AI Benchmark:** Mira a dispositivi mobili e consumer, valutando le prestazioni dell'IA nelle applicazioni per utenti finali (Ignatov et al. 2018).

Figura 10.4: MLPerf Training v3.0 e suoi utilizzi. Fonte: Forbes

I benchmark hanno anche metriche delle prestazioni che sono misure quantificabili utilizzate per valutare l'efficacia degli acceleratori di IA. Queste metriche forniscono una visione completa delle capacità di un acceleratore e vengono utilizzate per guidare il processo di progettazione e selezione per i sistemi di IA. Le metriche comuni comprendono:

- **Throughput:** Solitamente misurato in operazioni al secondo, questo parametro indica il volume di calcoli che un acceleratore può gestire.
- **Latenza:** Il ritardo temporale tra input e output in un sistema è fondamentale per le attività di elaborazione in tempo reale.
- **Efficienza Energetica:** Calcolato come elaborazione per watt, che rappresenta il compromesso tra prestazioni e consumo energetico.
- **Efficienza dei Costi:** Valuta il costo operativo in relazione alle prestazioni, un parametro essenziale per le distribuzioni attente al budget.
- **Precisione:** Nelle attività di inferenza, la precisione dei calcoli è fondamentale e talvolta bilanciata rispetto alla velocità.
- **Scalabilità:** La capacità del sistema di mantenere i guadagni in termini di prestazioni man mano che il carico computazionale aumenta.

I risultati del benchmark forniscono informazioni che vanno oltre i semplici numeri: possono rivelare colli di bottiglia nello stack software e nell'hardware. Ad esempio, i benchmark possono mostrare come l'aumento delle dimensioni del batch migliori l'utilizzo della GPU fornendo più parallelismo o come le ottimizzazioni del compilatore aumentino le prestazioni della TPU. Questi insegnamenti consentono un'ottimizzazione continua (Zhihao Jia, Zaharia, e Aiken 2019).

Il benchmarking standardizzato fornisce una valutazione quantificata e comparabile degli acceleratori AI per informare la progettazione, l'acquisto e l'ottimizzazione. Tuttavia, anche la convalida delle prestazioni nel mondo reale rimane essenziale (H. Zhu et al. 2018).

10.7 Sfide e Soluzioni

Gli acceleratori AI offrono notevoli miglioramenti delle prestazioni, ma spesso è necessario migliorare i significativi problemi di portabilità e compatibilità nella loro integrazione nel più ampio panorama AI. Il nocciolo della questione risiede nella diversità dell'ecosistema AI:

esiste una vasta gamma di acceleratori, framework e linguaggi di programmazione per l'apprendimento automatico, ognuno con le sue caratteristiche e requisiti unici.

10.7.1 Problemi di Portabilità/Compatibilità

Gli sviluppatori incontrano spesso difficoltà nel trasferire i loro modelli AI da un ambiente hardware a un altro. Ad esempio, un modello di machine learning sviluppato per un ambiente desktop in Python utilizzando il framework PyTorch, ottimizzato per una GPU Nvidia, potrebbe non essere facilmente trasferito a un dispositivo più vincolato come Arduino Nano 33 BLE. Questa complessità deriva da nette differenze nei requisiti di programmazione: Python e PyTorch sul desktop rispetto a un ambiente C++ su un Arduino, per non parlare del passaggio dall'architettura x86 ad ARM ISA.
Queste divergenze evidenziano la complessità della portabilità all'interno dei sistemi AI. Inoltre, il rapido progresso negli algoritmi e nei modelli di intelligenza artificiale implica che gli acceleratori hardware debbano adattarsi continuamente, creando un obiettivo mobile per la compatibilità. L'assenza di standard e interfacce universali aggrava il problema, rendendo difficile l'implementazione di soluzioni di intelligenza artificiale in modo coerente su vari dispositivi e piattaforme.

10.7.1.1 Soluzioni e Strategie

Per affrontare questi ostacoli, il settore dell'intelligenza artificiale si sta muovendo verso diverse soluzioni:

10.7.1.1.1 Iniziative di Standardizzazione

Open Neural Network Exchange (ONNX) è in prima linea in questa ricerca, proponendo un ecosistema aperto e condiviso che promuove l'intercambiabilità dei modelli. ONNX facilita l'uso di modelli di intelligenza artificiale su vari framework, consentendo ai modelli addestrati in un ambiente di essere distribuiti in modo efficiente in un altro, riducendo significativamente la necessità di riscritture o modifiche che richiedono molto tempo.

10.7.1.1.2 Framework Multipiattaforma

A complemento degli sforzi di standardizzazione, framework multipiattaforma come TensorFlow Lite e PyTorch Mobile sono stati sviluppati specificamente per creare coesione tra diversi ambienti di calcolo che vanno dai desktop ai dispositivi mobili ed embedded. Questi framework offrono versioni semplificate e leggere delle loro versioni principali, garantendo compatibilità e integrità funzionale su diversi tipi di hardware senza sacrificare le prestazioni. Ciò garantisce che gli sviluppatori possano creare applicazioni con la certezza che funzioneranno su molti dispositivi, colmando un divario che tradizionalmente ha rappresentato una sfida considerevole nello sviluppo dell'intelligenza artificiale.

10.7.1.1.3 Piattaforme Indipendenti dall'Hardware

L'ascesa delle piattaforme indipendenti dall'hardware ha anche svolto un ruolo importante nella democratizzazione dell'uso dell'IA. Creando ambienti in cui le applicazioni di IA possono essere eseguite su vari acceleratori, queste piattaforme eliminano l'onere della codifica specifica per l'hardware dagli sviluppatori. Questa astrazione semplifica il processo di sviluppo e apre nuove possibilità per l'innovazione e l'implementazione delle applicazioni, libere dai vincoli delle specifiche hardware.

10.7.1.1.4 Strumenti di Compilazione Avanzati

Inoltre, l'avvento di strumenti di compilazione avanzati come TVM, un compilatore di tensori end-to-end, offre un percorso ottimizzato attraverso la giungla delle diverse architetture hardware. TVM fornisce agli sviluppatori i mezzi per mettere a punto modelli di machine learning per un ampio spettro di substrati computazionali, garantendo prestazioni ottimali ed

evitando la regolazione manuale del modello ogni volta che si verifica uno spostamento nell'hardware sottostante.

10.7.1.1.5 Collaborazione tra Comunità e Settore

La collaborazione tra comunità open source e consorzi di settore non può essere sottovalutata. Questi organismi collettivi sono fondamentali per la formazione di standard condivisi e best practice a cui tutti gli sviluppatori e i produttori possono aderire. Tale collaborazione promuove un ecosistema AI più unificato e sinergico, riducendo significativamente la prevalenza di problemi di portabilità e spianando la strada verso l'integrazione e l'avanzamento dell'AI globale. Attraverso questi lavori combinati, l'AI si sta muovendo costantemente verso un futuro in cui la distribuzione di modelli senza soluzione di continuità su varie piattaforme diventa uno standard piuttosto che un'eccezione.

Risolvere le sfide della portabilità è fondamentale per il campo dell'IA per realizzare il pieno potenziale degli acceleratori hardware in un panorama tecnologico dinamico e diversificato. Richiede uno sforzo concertato da parte dei produttori di hardware, degli sviluppatori di software e degli enti normativi per creare un ambiente più interoperabile e flessibile. Con innovazione e collaborazione continue, la comunità dell'IA può aprire la strada a un'integrazione e a un'implementazione senza soluzione di continuità dei modelli di IA su molte piattaforme.

10.7.2 Problemi di Consumo Energetico

Il consumo energetico è un problema cruciale nello sviluppo e nel funzionamento degli acceleratori AI dei data center, come le unità di elaborazione grafica (GPU) e le unità di elaborazione tensoriale (TPU) (N. P. Jouppi et al. 2017b) (Norrie et al. 2021) (N. Jouppi et al. 2023). Questi potenti componenti sono la spina dorsale dell'infrastruttura AI contemporanea, ma le loro elevate richieste di energia contribuiscono all'impatto ambientale della tecnologia e aumentano significativamente i costi operativi. Man mano che le esigenze di elaborazione dei dati diventano più complesse, con la crescente popolarità dell'AI e del deep learning, c'è una richiesta pressante di GPU e TPU in grado di fornire la potenza di calcolo necessaria in modo più efficiente. L'impatto di tali progressi è duplice: possono ridurre l'impatto ambientale di queste tecnologie e ridurre i costi di esecuzione delle applicazioni AI.

Le tecnologie hardware emergenti sono sul punto di rivoluzionare l'efficienza energetica in questo settore. L'informatica fotonica, ad esempio, utilizza la luce anziché l'elettricità per trasportare informazioni, offrendo la promessa di un'elaborazione ad alta velocità con una frazione del consumo energetico. Analizziamo più approfonditamente questa e altre tecnologie innovative nella sezione "Tecnologie Hardware Emergenti", esplorando il loro potenziale per affrontare le attuali sfide del consumo energetico.

Ai margini della rete, gli acceleratori AI sono progettati per elaborare dati su dispositivi come smartphone, sensori IoT e dispositivi indossabili intelligenti. Questi dispositivi spesso funzionano con gravi limitazioni di potenza, rendendo necessario un attento bilanciamento tra prestazioni e consumo energetico. Un modello AI ad alte prestazioni può fornire risultati rapidi, ma a costo di esaurire rapidamente la durata della batteria e aumentare la produzione termica, il che può influire sulla funzionalità e sulla durata del dispositivo. La posta in gioco è più alta per i dispositivi distribuiti in aree remote o difficili da raggiungere, dove non è possibile garantire un'alimentazione costante, il che sottolinea la necessità di soluzioni a basso consumo energetico.

I problemi di latenza aggravano ulteriormente la sfida dell'efficienza energetica ai margini. Le applicazioni AI Edge in settori quali la guida autonoma e il monitoraggio sanitario richiedono velocità, precisione e affidabilità, poiché i ritardi nell'elaborazione possono comportare gravi rischi per la sicurezza. Per queste applicazioni, gli sviluppatori devono ottimizzare sia gli

algoritmi AI sia la progettazione hardware per raggiungere un equilibrio ottimale tra consumo energetico e latenza.
Questo sforzo di ottimizzazione non riguarda solo l'apporto di miglioramenti incrementali alle tecnologie esistenti; riguarda il ripensamento di come e dove elaboriamo le attività AI. Progettando acceleratori AI che siano sia efficienti dal punto di vista energetico sia in grado di elaborare rapidamente, possiamo garantire che questi dispositivi svolgano i loro scopi previsti senza un consumo energetico non necessario o prestazioni compromesse. Tali sviluppi potrebbero promuovere l'adozione diffusa dell'AI in vari settori, consentendo un uso più intelligente, sicuro e sostenibile della tecnologia.

10.7.3 Superare i Vincoli delle Risorse

Anche i vincoli di risorse rappresentano una sfida significativa per gli acceleratori Edge AI, poiché queste soluzioni hardware e software specializzate devono fornire prestazioni robuste entro i limiti dei dispositivi edge. A causa dei limiti di potenza e dimensioni, gli acceleratori Edge AI hanno spesso capacità di calcolo, memoria e archiviazione limitate (L. Zhu et al. 2023). Questa scarsità di risorse richiede un'attenta allocazione delle capacità di elaborazione per eseguire modelli di apprendimento automatico in modo efficiente.
Inoltre, la gestione di risorse limitate richiede approcci innovativi, tra cui la quantizzazione del modello (Lin et al. 2023) (Y. Li, Dong, e Wang 2020), pruning (T. Wang et al. 2020) e l'ottimizzazione delle pipeline di inferenza. Gli acceleratori Edge AI devono trovare un delicato equilibrio tra la fornitura di funzionalità AI significative e il non esaurire le risorse disponibili, mantenendo al contempo un basso consumo energetico. Superare questi vincoli di risorse è fondamentale per garantire l'implementazione di successo dell'intelligenza artificiale ai margini, dove molte applicazioni, dall'IoT ai dispositivi mobili, si basano sull'uso efficiente di risorse hardware limitate per fornire un processo decisionale intelligente e in tempo reale.

10.8 Tecnologie Emergenti

Finora abbiamo discusso la tecnologia hardware AI nel contesto della progettazione dell'architettura von Neumann convenzionale e dell'implementazione basata su CMOS. Questi chip AI specializzati offrono vantaggi come una maggiore produttività ed efficienza energetica, ma si basano sui principi di elaborazione tradizionali. La crescita inarrestabile della domanda di potenza di elaborazione AI sta guidando le innovazioni nei metodi di integrazione per l'hardware AI.
Sono emersi due approcci principali per massimizzare la densità di elaborazione, l'integrazione su "scala wafer" e le architetture basate su "chiplet", di cui parleremo in questa sezione. Guardando molto più avanti, esamineremo le tecnologie emergenti che divergono dalle architetture convenzionali e adottano approcci fondamentalmente diversi per l'elaborazione specializzata AI.
Alcuni di questi paradigmi non convenzionali includono l'elaborazione neuromorfica, che imita le reti neurali biologiche; l'elaborazione quantistica, che sfrutta gli effetti della meccanica quantistica; e l'elaborazione ottica, che utilizza fotoni anziché elettroni. Oltre ai nuovi substrati di elaborazione, le nuove tecnologie dei dispositivi stanno consentendo ulteriori guadagni attraverso una migliore memoria e interconnessione.
Esempi includono i "memristor" [https://it.wikipedia.org/wiki/Memristore] per l'elaborazione in memoria e la nanofotonica per la comunicazione fotonica integrata. Insieme, queste tecnologie offrono il potenziale per miglioramenti di ordini di grandezza in termini di velocità, efficienza e scalabilità rispetto all'attuale hardware AI. Esamineremo questi aspetti in questa sezione.

10.8.1 Metodi di Integrazione

I metodi di integrazione si riferiscono agli approcci utilizzati per combinare e interconnettere i vari componenti di elaborazione e memoria di un chip o sistema AI. Collegando strettamente gli elementi di elaborazione chiave, l'integrazione cerca di massimizzare le prestazioni, l'efficienza energetica e la densità.

In passato, l'elaborazione AI veniva eseguita principalmente su CPU e GPU costruite utilizzando metodi di integrazione convenzionali. Questi componenti discreti venivano fabbricati separatamente e collegati insieme su una scheda. Tuttavia, questa integrazione poco stretta crea colli di bottiglia, come i sovraccarichi dei trasferimento di dati.

Con l'aumento dei carichi di lavoro AI, aumenta la domanda di una più stretta integrazione tra elementi di elaborazione, memoria e comunicazione. Alcuni fattori chiave dell'integrazione includono:

- **Riduzione al minimo dello spostamento dei dati:** Una stretta integrazione riduce la latenza e l'energia per lo spostamento dei dati tra i componenti. Ciò migliora l'efficienza.
- **Personalizzazione:** Adattare tutti i componenti del sistema ai carichi di lavoro AI consente ottimizzazioni in tutto lo stack hardware.
- **Parallelismo:** L'integrazione di molti elementi di elaborazione consente un calcolo parallelo massiccio.
- **Densità:** Una più stretta integrazione consente di impacchettare più transistor e memoria in una determinata area.
- **Costo:** Le economie di scala derivanti da grandi sistemi integrati possono ridurre i costi.

In risposta, nuove tecniche di produzione come la fabbricazione su scala di wafer e il confezionamento avanzato consentono ora livelli di integrazione molto più elevati. L'obiettivo è creare complessi di elaborazione AI unificati e specializzati, su misura per il deep learning e altri algoritmi AI. Un'integrazione più stretta è fondamentale per fornire le prestazioni e l'efficienza necessarie per la prossima generazione di AI.

10.8.1.1 AI su Scala Wafer

L'intelligenza artificiale su "wafer-scale" adotta un approccio estremamente integrato, producendo un intero wafer di silicio come un gigantesco chip. Ciò differisce drasticamente dalle CPU e GPU convenzionali, che tagliano ogni wafer in molti chip singoli più piccoli. Figura 10.5 mostra un confronto tra Cerebras Wafer Scale Engine 2, che è il chip più grande mai costruito, e la GPU più grande. Mentre alcune GPU possono contenere miliardi di transistor, impallidiscono comunque rispetto alla scala di un chip delle dimensioni di un wafer con oltre un trilione di transistor.

Figura 10.5: Wafer-scale vs. GPU. Fonte: Cerebras.

L'approccio su scala di wafer diverge anche dai progetti system-on-chip più modulari che hanno ancora componenti discreti che comunicano tramite bus. Invece, l'intelligenza artificiale su scala di wafer consente la personalizzazione completa e la stretta integrazione di elaborazione, memoria e interconnessioni nell'intero di die.
Progettando il wafer come un'unità logica integrata, il trasferimento dati tra gli elementi è ridotto al minimo. Ciò fornisce una latenza e un consumo energetico inferiori rispetto ai design discreti system-on-chip o chiplet. Mentre i chiplet possono offrire flessibilità mescolando e abbinando i componenti, la comunicazione tra chiplet è impegnativa. La natura monolitica dell'integrazione su scala wafer elimina questi colli di bottiglia nella comunicazione tra chip.
Tuttavia, la scala ultra-large pone anche difficoltà per la producibilità e la resa con i design su scala wafer. Difetti in qualsiasi regione del wafer possono rendere (alcune parti del) chip inutilizzabile. Sono necessarie tecniche di litografia specializzate per produrre tali matrici di grandi dimensioni. Quindi, l'integrazione su scala wafer persegue i massimi guadagni in termini di prestazioni dall'integrazione ma richiede il superamento di sostanziali sfide di fabbricazione.
Video 10.1 fornisce ulteriore contesto sui chip AI su scala wafer.

Video 10.1: Wafer-scale AI Chips

https://www.youtube.com/watch?v=Fcob512SJz0

10.8.1.2 Chiplet per AI

Il design chiplet si riferisce a un'architettura semiconduttrice in cui un singolo circuito integrato (IC) è costruito da più componenti più piccoli e individuali noti come chiplet. Ogni chiplet è un blocco funzionale autonomo, in genere specializzato per un'attività o funzionalità specifica. Questi chiplet sono quindi interconnessi su un substrato o un package più grande per creare un sistema coeso.
Figura 10.6 illustra questo concetto. Per l'hardware AI, i chiplet consentono di combinare diversi tipi di chip ottimizzati per attività come moltiplicazione di matrici, spostamento di dati, I/O analogico e memorie specializzate. Questa integrazione eterogenea differisce

notevolmente dall'integrazione wafer-scale, in cui tutta la logica è prodotta come un unico chip monolitico. Aziende come Intel e AMD hanno adottato design chiplet per le loro CPU.
I chiplet sono interconnessi utilizzando tecniche di packaging avanzate come interposer di substrato ad alta densità, impilamento 2.5D/3D e packaging a livello di wafer. Ciò consente di combinare chiplet realizzati con diversi nodi di processo, memorie specializzate e vari motori AI ottimizzati.

Figura 10.6: Partizionamento chiplet. Fonte: Vivet et al. (2021).

Ecco alcuni vantaggi chiave dell'uso di chiplet per l'intelligenza artificiale:

- **Flessibilità:** I chiplet consentono la combinazione di diversi tipi di chip, nodi di processo e memorie su misura per ogni funzione. Questo è più modulare rispetto a un design fisso su scala wafer.
- **Resa:** I chiplet più piccoli hanno una resa maggiore rispetto a un gigantesco chip su scala wafer. I difetti sono contenuti nei singoli chiplet.
- **Costo:** Sfrutta le capacità di produzione esistenti anziché richiedere nuovi processi specializzati. Riduce i costi riutilizzando la fabbricazione assestata.
- **Compatibilità:** Può integrarsi con architetture di sistema più convenzionali come PCIe e interfacce di memoria DDR standard.

Tuttavia, i chiplet devono anche affrontare sfide di integrazione e prestazioni:

- Densità inferiore rispetto alla scala wafer, poiché i chiplet sono limitati in termini di dimensioni.
- Latenza aggiuntiva durante la comunicazione tra chiplet rispetto all'integrazione monolitica. Richiede ottimizzazione per interconnessioni a bassa latenza.
- Il packaging avanzato aggiunge complessità rispetto all'integrazione su scala wafer, sebbene ciò sia discutibile.

L'obiettivo principale dei chiplet è trovare il giusto equilibrio tra flessibilità modulare e densità di integrazione per prestazioni AI ottimali. I chiplet mirano a un'accelerazione AI efficiente pur lavorando entro i vincoli delle tecniche di produzione convenzionali. I chiplet prendono una via di mezzo tra gli estremi dell'integrazione su scala wafer e dei componenti completamente discreti. Ciò fornisce vantaggi pratici ma può sacrificare una certa densità computazionale ed efficienza rispetto a un sistema teorico a livello di wafer.

10.8.2 Elaborazione Nùeuromorfica

L'elaborazione neuromorfica è un campo emergente che mira a emulare l'efficienza e la robustezza dei sistemi neurali biologici per applicazioni di machine learning. Una differenza

fondamentale rispetto alle classiche architetture di Von Neumann è la fusione di memoria ed elaborazione nello stesso circuito (Schuman et al. 2022; Marković et al. 2020; Furber 2016), come illustrato in Figura 10.7. La struttura del cervello ispira questo approccio integrato. Un vantaggio fondamentale è il potenziale per un miglioramento di ordini di grandezza nel calcolo efficiente dal punto di vista energetico rispetto all'hardware AI convenzionale. Ad esempio, le stime prevedono guadagni di 100x-1000x nell'efficienza energetica rispetto agli attuali sistemi basati su GPU per carichi di lavoro equivalenti.

Figura 10.7: Confronto tra l'architettura di von Neumann e l'architettura neuromorfica. Fonte: Schuman et al. (2022).

Intel e IBM stanno guidando gli sforzi commerciali nell'hardware neuromorfico. I chip Loihi e Loihi 2 di Intel (M. Davies et al. 2018, 2021) offrono core neuromorfici programmabili con apprendimento on-chip. Il dispositivo Northpole (Modha et al. 2023) di IBM comprende oltre 100 milioni di sinapsi a giunzione a tunnel magnetico e 68 miliardi di transistor. Questi chip specializzati offrono vantaggi come un basso consumo energetico per l'inferenza edge.
Le "Spiking neural network (SNN)" (Maass 1997) sono modelli computazionali per hardware neuromorfico. A differenza delle reti neurali profonde che comunicano tramite valori continui, le SNN utilizzano picchi discreti che sono più simili ai neuroni biologici. Questo consente un calcolo efficiente basato sugli eventi anziché un'elaborazione costante. Inoltre, le SNN considerano le caratteristiche temporali e spaziali dei dati di input. Ciò imita meglio le reti neurali biologiche, in cui la tempistica dei picchi neuronali svolge un ruolo importante. Tuttavia, l'addestramento delle SNN rimane impegnativo a causa della complessità temporale aggiunta. Figura 10.8 fornisce una panoramica della metodologia spiking: (a) illustrazione di un neurone; (b) Misura di un potenziale d'azione propagato lungo l'assone di un neurone. Solo il potenziale d'azione è rilevabile lungo l'assone; (c) Il picco del neurone è approssimato con una rappresentazione binaria; (d) Elaborazione guidata dagli eventi; (e) Active Pixel Sensor e Dynamic Vision Sensor.

Figura 10.8: Spiking neuromorfico. Fonte: Eshraghian et al. (2023).

Si può anche guardare Video 10.2 linkato di seguito per una spiegazione più dettagliata.

> Video 10.2: Neuromorphic Computing
>
> https://www.youtube.com/watch?v=yihk_8XnCzg

Dispositivi nanoelettronici specializzati chiamati memristor (Chua 1971) sono componenti sinaptici nei sistemi neuromorfici. I memristor agiscono come memoria non volatile con conduttanza regolabile, emulando la plasticità delle sinapsi reali. I memristor consentono l'apprendimento in situ senza trasferimenti di dati separati combinando funzioni di memoria ed elaborazione. Tuttavia, la tecnologia dei memristor deve ancora raggiungere la maturità e la scalabilità per l'hardware commerciale.

L'integrazione della fotonica con il calcolo neuromorfico (Shastri et al. 2021) è emersa di recente come un'area di ricerca attiva. L'uso della luce per il calcolo e la comunicazione consente alte velocità e un consumo energetico ridotto. Tuttavia, la piena realizzazione di sistemi neuromorfici fotonici richiede il superamento di problemi di progettazione e integrazione.

Il calcolo neuromorfico offre promettenti capacità per un'efficace inferenza edge, ma incontra ostacoli in merito ad algoritmi di addestramento, integrazione dei nanodispositivi e progettazione del sistema. La ricerca multidisciplinare in corso in informatica, ingegneria, scienza dei materiali e fisica sarà fondamentale per sbloccare il pieno potenziale di questa tecnologia per i casi d'uso dell'intelligenza artificiale.

10.8.3 Calcolo Analogico

Il computing analogico è un approccio emergente che utilizza segnali e componenti analogici come condensatori, induttori e amplificatori anziché la logica digitale per il calcolo. Rappresenta le informazioni come segnali elettrici continui anziché 0 e 1 discreti. Ciò consente al calcolo di riflettere direttamente la natura analogica dei dati del mondo reale, evitando errori di digitalizzazione e overhead.

Il computing analogico ha generato un rinnovato interesse per l'hardware AI efficiente, in particolare per l'inferenza direttamente su dispositivi edge a basso consumo. I circuiti analogici, come la moltiplicazione e la sommatoria al centro delle reti neurali, possono

essere utilizzati con un consumo energetico molto basso. Ciò rende l'analogico adatto per l'implementazione di modelli ML su nodi finali con vincoli energetici. Startup come Mythic stanno sviluppando acceleratori AI analogici.

Mentre il computing analogico era popolare nei primi computer, il boom della logica digitale ha portato al suo declino. Tuttavia, l'analogico è convincente per applicazioni di nicchia che richiedono estrema efficienza (Haensch, Gokmen, e Puri 2019). Contrasta con gli approcci neuromorfici digitali che utilizzano ancora picchi digitali per il calcolo. L'analogico può consentire un calcolo di precisione inferiore, ma richiede competenza nella progettazione di circuiti analogici. I compromessi su precisione, complessità di programmazione e costi di fabbricazione rimangono aree di ricerca attive.

Il calcolo neuromorfico, che emula i sistemi neurali biologici per un'inferenza ML efficiente, può utilizzare circuiti analogici per implementare i componenti e i comportamenti chiave del cervello. Ad esempio, i ricercatori hanno progettato circuiti analogici per modellare neuroni e sinapsi utilizzando condensatori, transistor e amplificatori operazionali (Hazan e Ezra Tsur 2021). I condensatori possono esibire le dinamiche di picco dei neuroni biologici, mentre gli amplificatori e i transistor forniscono una somma ponderata di input per imitare i dendriti. Le tecnologie a resistore variabile come i memristor possono realizzare sinapsi analogiche con plasticità dipendente dal tempo di picco, che può rafforzare o indebolire le connessioni in base all'attività di picco.

Startup come SynSense hanno sviluppato chip neuromorfici analogici contenenti questi componenti biomimetici (Bains 2020). Questo approccio analogico si traduce in un basso consumo energetico e un'elevata scalabilità per i dispositivi edge rispetto alle complesse implementazioni SNN digitali.

Tuttavia, l'addestramento di SNN analogiche sui chip rimane una sfida aperta. Nel complesso, la realizzazione analogica è una tecnica promettente per fornire l'efficienza, la scalabilità e la plausibilità biologica previste con il calcolo neuromorfico. La fisica dei componenti analogici combinata con la progettazione dell'architettura neurale potrebbe migliorare l'efficienza dell'inferenza rispetto alle reti neurali digitali convenzionali.

10.8.4 Elettronica Flessibile

Mentre gran parte della nuova tecnologia hardware nell'area di lavoro ML si è concentrata sull'ottimizzazione e sulla creazione di sistemi più efficienti, c'è una traiettoria parallela che mira ad adattare l'hardware per applicazioni specifiche (Gates 2009; Musk et al. 2019; Tang et al. 2023; Tang, He, e Liu 2022; S. H. Kwon e Dong 2022). Una di queste strade è lo sviluppo di elettronica flessibile per casi d'uso AI.

L'elettronica flessibile si riferisce a circuiti elettronici e dispositivi fabbricati su substrati flessibili in plastica o polimeri anziché in silicio rigido. A differenza delle schede e dei chip rigidi convenzionali, ciò consente all'elettronica di piegarsi, torcersi e adattarsi a forme irregolari. Figura 10.9 mostra un esempio di un prototipo di dispositivo flessibile che misura in modalità wireless la temperatura corporea, che può essere integrato senza soluzione di continuità in indumenti o cerotti cutanei. La flessibilità e la piegabilità dei materiali elettronici emergenti consentono di integrarli in fattori di forma sottili e leggeri, adatti per applicazioni AI e TinyML embedded.

L'hardware AI flessibile può adattarsi a superfici curve e funzionare in modo efficiente con budget di potenza in microwatt. La flessibilità consente inoltre fattori di forma arrotolabili o pieghevoli per ridurre al minimo l'ingombro e il peso del dispositivo, ideali per piccoli dispositivi intelligenti portatili e dispositivi indossabili che incorporano TinyML. Un altro vantaggio fondamentale dell'elettronica flessibile rispetto alle tecnologie convenzionali sono i costi di produzione inferiori e i processi di fabbricazione più semplici, che potrebbero democratizzare l'accesso a queste tecnologie. Mentre le maschere in silicio e i costi di

fabbricazione in genere costano milioni di dollari, l'hardware flessibile in genere costa solo decine di centesimi per la produzione (Huang et al. 2011; Biggs et al. 2021). Il potenziale di fabbricare elettronica flessibile direttamente su pellicole di plastica utilizzando processi di stampa e rivestimento ad alta produttività può ridurre i costi e migliorare la producibilità su larga scala rispetto ai chip AI rigidi (Musk et al. 2019).

Figura 10.9: Prototipo di dispositivo flessibile. Fonte: Jabil Circuit.

Il campo è abilitato dai progressi nei semiconduttori organici e nei nanomateriali che possono essere depositati su pellicole sottili e flessibili. Tuttavia, la fabbricazione rimane impegnativa rispetto ai processi maturi del silicio. I circuiti flessibili attualmente presentano in genere prestazioni inferiori rispetto agli equivalenti rigidi. Tuttavia, promettono di trasformare l'elettronica in materiali leggeri e pieghevoli.

I casi d'uso dell'elettronica flessibile sono adatti per l'integrazione intima con il corpo umano. Le potenziali applicazioni dell'intelligenza artificiale medica includono sensori biointegrati, "soft robot" e impianti che monitorano o stimolano il sistema nervoso in modo intelligente. In particolare, gli array di elettrodi flessibili potrebbero consentire interfacce neurali a densità più elevata e meno invasive rispetto agli equivalenti rigidi.

Pertanto, l'elettronica flessibile sta inaugurando una nuova era di dispositivi indossabili e sensori corporei, in gran parte grazie alle innovazioni nei transistor organici. Questi componenti consentono un'elettronica più leggera e pieghevole, ideale per dispositivi indossabili, pelle elettronica e dispositivi medici che si adattano al corpo.

Sono adatti per dispositivi bioelettronici in termini di biocompatibilità, aprendo la strada ad applicazioni in interfacce cerebrali e cardiache. Ad esempio, la ricerca sulle interfacce flessibili cervello-computer e sulla bioelettronica morbida per applicazioni cardiache dimostra il potenziale per applicazioni mediche di vasta portata.

Aziende e istituti di ricerca non stanno solo sviluppando e investendo grandi quantità di risorse in elettrodi flessibili, come mostrato nel lavoro di Neuralink (Musk et al. 2019). Tuttavia, stanno anche spingendo i confini per integrare modelli di apprendimento automatico nei sistemi (S. H. Kwon e Dong 2022). Questi sensori intelligenti mirano a una simbiosi fluida e duratura con il corpo umano.

Eticamente, l'incorporazione di sensori intelligenti basati sull'apprendimento automatico nel corpo solleva importanti questioni. Le problematiche relative alla privacy dei dati, al consenso informato e alle implicazioni sociali a lungo termine di tali tecnologie sono al centro del lavoro in corso in neuroetica e bioetica (Segura Anaya et al. 2017; Goodyear 2017;

Farah 2005; Roskies 2002). Il campo sta progredendo a un ritmo che richiede progressi paralleli nei parametri etici per guidare lo sviluppo e l'implementazione responsabili di queste tecnologie. Sebbene vi siano limitazioni e ostacoli etici da superare, le prospettive per l'elettronica flessibile sono ampie e promettono molto per la ricerca e le applicazioni future.

10.8.5 Tecnologie delle Memorie

Le tecnologie delle memorie sono fondamentali per l'hardware AI, ma la DDR DRAM e la SRAM convenzionali creano colli di bottiglia. I carichi di lavoro AI richiedono un'elevata larghezza di banda (>1 TB/s). Le applicazioni scientifiche estreme dell'AI richiedono una latenza estremamente bassa (<50 ns) per alimentare i dati alle unità di calcolo (Duarte et al. 2022), un'elevata densità (>128 Gb) per archiviare grandi parametri di modelli e set di dati e un'eccellente efficienza energetica (<100 fJ/b) per uso embedded (N. Verma et al. 2019). Sono necessarie nuove memorie per soddisfare queste esigenze. Le opzioni emergenti includono diverse nuove tecnologie:

- La RAM resistiva (ReRAM) può migliorare la densità con semplici array passivi. Tuttavia, permangono dei problemi legati alla variabilità (Chi et al. 2016).
- La "Phase change memory (PCM)" [memoria a cambiamento di fase] sfrutta le proprietà uniche del vetro calcogenuro. Le fasi cristalline e amorfe hanno resistenze diverse. L'Optane DCPMM di Intel fornisce PCM veloci (100 ns) e ad alta resistenza. Tuttavia, le sfide includono cicli di scrittura limitati e corrente di reset elevata (Burr et al. 2016).
- Lo stacking 3D può anche aumentare la densità di memoria e la larghezza di banda integrando verticalmente strati di memoria con interconnessioni TSV (Loh 2008). Ad esempio, HBM fornisce interfacce larghe 1024 bit.

Le nuove tecnologie di memoria, con le loro innovative architetture e materiali cellulari, sono fondamentali per sbloccare il prossimo livello di prestazioni ed efficienza hardware AI. Realizzare i loro vantaggi nei sistemi commerciali rimane una sfida continua.
L'elaborazione in-memory sta guadagnando terreno come promettente strada per ottimizzare l'apprendimento automatico e i carichi di lavoro di elaborazione ad alte prestazioni. Al centro, la tecnologia colloca l'archiviazione e l'elaborazione dei dati per migliorare l'efficienza energetica e ridurre la latenza Wong et al. (2012). Due tecnologie chiave sotto questo ombrello sono la "Resistive RAM (ReRAM)" e il "Processing-In-Memory (PIM)". ReRAM (Wong et al. 2012) e PIM (Chi et al. 2016) sono le colonne portanti per l'elaborazione in memoria, l'archiviazione e l'elaborazione dei dati nella stessa posizione. ReRAM si concentra su questioni di uniformità, resistenza, conservazione, funzionamento multi-bit e scalabilità. D'altro canto, PIM coinvolge unità CPU integrate direttamente in array di memoria, specializzate per attività come la moltiplicazione di matrici, che sono centrali nei calcoli AI.
Queste tecnologie trovano applicazioni nei carichi di lavoro AI e nell'elaborazione ad alte prestazioni, dove la sinergia di storage e calcolo può portare a significativi guadagni in termini di prestazioni. L'architettura è particolarmente utile per le attività di elaborazione intensiva comuni nei modelli di apprendimento automatico.
Mentre le tecnologie di elaborazione in memoria come ReRAM e PIM offrono interessanti prospettive di efficienza e prestazioni, presentano le loro sfide, come l'uniformità dei dati e i problemi di scalabilità in ReRAM (Imani, Rahimi, e S. Rosing 2016). Tuttavia, il campo è maturo per l'innovazione e affrontare queste limitazioni può aprire nuove frontiere nell'AI e nell'elaborazione ad alte prestazioni.

10.8.6 Calcolo Ottico

Nell'accelerazione dell'intelligenza artificiale, un'area di interesse in rapida crescita risiede nelle nuove tecnologie che si discostano dai paradigmi tradizionali. Alcune tecnologie emergenti menzionate sopra, come l'elettronica flessibile, il calcolo in memoria o persino il calcolo neuromorfico, stanno per diventare realtà, date le loro innovazioni e applicazioni rivoluzionarie. Una delle frontiere promettenti e all'avanguardia della prossima generazione è la tecnologia del calcolo ottico H. Zhou et al. (2022). Aziende come LightMatter sono pioniere nell'uso della fotonica luminosa per i calcoli, utilizzando quindi fotoni al posto degli elettroni per la trasmissione dei dati e l'elaborazione.

Il calcolo ottico utilizza fotoni e dispositivi fotonici anziché i tradizionali circuiti elettronici per il calcolo e l'elaborazione dei dati. Trae ispirazione dai collegamenti di comunicazione in fibra ottica che si basano sulla luce per un trasferimento dati rapido ed efficiente (Shastri et al. 2021). La luce può propagarsi con una perdita molto inferiore rispetto agli elettroni dei semiconduttori, consentendo vantaggi intrinseci in termini di velocità ed efficienza.

Alcuni vantaggi specifici dell'elaborazione ottica includono:

- **Alta produttività:** I fotoni possono trasmettere con larghezze di banda >100 Tb/s utilizzando il multiplexing a divisione di lunghezza d'onda.
- **Bassa latenza:** I fotoni interagiscono su scale temporali di femtosecondi, milioni di volte più velocemente dei transistor al silicio.
- **Parallelismo:** Più segnali di dati possono propagarsi simultaneamente attraverso lo stesso mezzo ottico.
- **Bassa potenza:** I circuiti fotonici che utilizzano guide d'onda e risonatori possono ottenere una logica e una memoria complesse con solo microwatt di potenza.

Tuttavia, l'elaborazione ottica deve attualmente affrontare sfide significative:

- Mancanza di memoria ottica equivalente alla RAM elettronica
- Richiede la conversione tra domini ottici ed elettrici.
- Set limitato di componenti ottici disponibili rispetto al ricco ecosistema elettronico.
- Metodi di integrazione immaturi per combinare la fotonica con i tradizionali chip CMOS.
- Modelli di programmazione complessi richiesti per gestire il parallelismo.

Di conseguenza, l'elaborazione ottica è ancora in una fase di ricerca molto precoce nonostante il suo potenziale promettente. Tuttavia, le innovazioni tecniche potrebbero consentirgli di integrare l'elettronica e sbloccare guadagni di prestazioni per i carichi di lavoro AI. Aziende come Lightmatter sono pioniere nei primi acceleratori ottici AI. A lungo termine, se le sfide chiave saranno superate, potrebbe rappresentare un substrato di elaborazione rivoluzionario.

10.8.7 Quantum Computing

I computer quantistici sfruttano fenomeni unici della fisica quantistica, come la sovrapposizione e l'entanglement, per rappresentare ed elaborare informazioni in modi non possibili in modo classico. Invece dei bit binari, l'unità fondamentale è il bit quantistico o qubit. A differenza dei bit classici, che sono limitati a 0 o 1, i qubit possono esistere simultaneamente in una sovrapposizione di entrambi gli stati a causa degli effetti quantistici. Anche più qubit possono essere entangled, portando a una densità di informazioni esponenziale ma introducendo risultati probabilistici. La sovrapposizione consente il calcolo parallelo su tutti gli stati possibili, mentre l'entanglement consente correlazioni non locali tra qubit. Figura 10.10 trasmette visivamente le differenze tra i bit classici nell'informatica e i bit quantistici (qbit).

Figura 10.10: Qubit, i mattoni del calcolo quantistico. Fonte: Microsoft

Gli algoritmi quantistici manipolano attentamente questi effetti meccanici quantistici intrinseci per risolvere problemi come l'ottimizzazione o la ricerca in modo più efficiente rispetto alle loro controparti classiche in teoria.

- Training più rapido di reti neurali profonde sfruttando il parallelismo quantistico per operazioni di algebra lineare.
- Gli algoritmi ML quantistici efficienti sfruttano le capacità uniche dei qubit.
- Reti neurali quantistiche con effetti quantistici intrinseci integrati nell'architettura del modello.
- Ottimizzatori quantistici che sfruttano algoritmi di "annealing" quantistica o adiabatici per problemi di ottimizzazione combinatoria.

Tuttavia, gli stati quantistici sono fragili e soggetti a errori che richiedono protocolli di correzione degli errori. La natura non intuitiva della programmazione quantistica introduce anche sfide non presenti nell'informatica classica.

- I bit quantistici rumorosi e fragili sono difficili da scalare. Il più grande computer quantistico odierno ha meno di 1000 qubit.
- Insieme limitato di porte e circuiti quantistici disponibili rispetto alla programmazione classica.
- Mancanza di set di dati e benchmark per valutare l'apprendimento automatico quantistico in domini pratici.

Sebbene un vantaggio quantistico significativo per l'apprendimento automatico sia ancora lontano, la ricerca attiva presso aziende come D-Wave, Rigetti e IonQ sta facendo progredire l'ingegneria informatica quantistica e gli algoritmi quantistici. Le principali aziende tecnologiche come Google, IBM e Microsoft stanno esplorando attivamente l'informatica quantistica. Google ha recentemente annunciato un processore quantistico a 72 qubit chiamato Bristlecone e prevede di costruire un sistema quantistico commerciale a 49 qubit.

Microsoft ha anche un programma di ricerca attivo nell'informatica quantistica topologica e collabora con la startup quantistica IonQ
Le tecniche quantistiche potrebbero prima fare breccia nell'ottimizzazione prima di un'adozione più generalizzata dell'apprendimento automatico. La realizzazione del pieno potenziale dell'apprendimento automatico quantistico attende importanti traguardi nello sviluppo dell'hardware quantistico e nella maturità dell'ecosistema. Figura 10.11 confronta a titolo esemplificativo l'informatica quantistica e quella classica.

Figura 10.11: Confronto tra il calcolo quantistico e il calcolo classico. Fonte: Devopedia

10.9 Tendenze Future

In questo capitolo, l'attenzione principale è stata rivolta alla progettazione di hardware specializzato ottimizzato per carichi di lavoro e algoritmi di machine learning. Questa discussione ha riguardato le architetture personalizzate di GPU e TPU per l'addestramento e l'inferenza delle reti neurali. Tuttavia, una direzione di ricerca emergente sta sfruttando l'apprendimento automatico per facilitare il processo di progettazione hardware stesso.
Il processo di progettazione hardware comprende molte fasi complesse, tra cui specifica, modellazione di alto livello, simulazione, sintesi, verifica, prototipazione e fabbricazione. Gran parte di questo processo richiede tradizionalmente una vasta competenza umana, impegno e tempo. Tuttavia, i recenti progressi nell'apprendimento automatico stanno consentendo l'automazione e il miglioramento di parti del flusso di lavoro di progettazione hardware utilizzando tecniche di apprendimento automatico.
Ecco alcuni esempi di come l'apprendimento automatico sta trasformando la progettazione hardware:

- **Sintesi di circuiti automatizzata tramite apprendimento per rinforzo:** Anziché realizzare manualmente progetti a livello di transistor, gli agenti di apprendimento automatico come l'apprendimento per rinforzo possono imparare a collegare porte logiche e generare automaticamente layout di circuiti. Ciò può accelerare il lungo processo di sintesi.
- **Simulazione ed emulazione hardware basate su ML:** I modelli di reti neurali profonde possono essere addestrati per prevedere come si comporterà un progetto hardware in diverse condizioni. Ad esempio, i modelli di apprendimento profondo possono essere addestrati per prevedere i conteggi dei cicli per determinati carichi di lavoro. Ciò consente una simulazione più rapida e accurata rispetto alle simulazioni RTL tradizionali.
- **Pianificazione automatizzata dei chip mediante algoritmi ML:** La pianificazione dei chip comporta il posizionamento ottimale di diversi componenti su un die. Algoritmi evolutivi come quelli genetici e altri algoritmi ML come l'apprendimento per rinforzo vengono utilizzati per esplorare le opzioni di pianificazione. Ciò può migliorare significativamente i posizionamenti manuali di pianificazione in termini di tempi di consegna più rapidi e qualità dei posizionamenti.
- **Ottimizzazione dell'architettura basata su ML:** Le nuove architetture hardware, come quelle per gli acceleratori ML efficienti, possono essere generate e ottimizzate automaticamente tramite la ricerca nello spazio di progettazione architettonica. Gli algoritmi di apprendimento automatico possono cercare efficacemente ampi spazi di progettazione architettonica.

L'applicazione del ML all'automazione della progettazione hardware promette di rendere il processo più veloce, più economico e più efficiente. Apre possibilità di progettazione che richiederebbero più di una progettazione manuale. L'uso del ML nella progettazione hardware è un'area di ricerca attiva e di distribuzione precoce, e studieremo le tecniche coinvolte e il loro potenziale trasformativo.

10.9.1 ML per l'automazione della progettazione hardware

Una grande opportunità per l'apprendimento automatico nella progettazione hardware è l'automazione di parti del complesso e noioso flusso di lavoro di progettazione. Con "Hardware design automation (HDA)" ci si riferisce in generale all'uso di tecniche ML come l'apprendimento per rinforzo, algoritmi genetici e reti neurali per automatizzare attività come sintesi, verifica, floorplanning e altro. Ecco alcuni esempi di dove l'ML per HDA mostra una vera promessa:

- **Sintesi di circuiti automatizzata:** La sintesi di circuiti comporta la conversione di una descrizione di alto livello della logica desiderata in un'implementazione di netlist a livello di gate ottimizzata. Questo processo complesso ha molte considerazioni e compromessi di progettazione. Gli agenti ML possono essere addestrati tramite l'apprendimento per rinforzo G. Zhou e Anderson (2023) per esplorare lo spazio di progettazione e produrre automaticamente sintesi ottimizzate. Startup come Symbiotic EDA stanno portando questa tecnologia sul mercato.
- **Automated chip floorplanning:** Il Floorplanning si riferisce al posizionamento strategico di diversi componenti su un'area del chip. Algoritmi di ricerca come algoritmi genetici (Valenzuela e Wang 2000) e apprendimento per rinforzo (Mirhoseini et al. (2021), Agnesina et al. (2023)) possono essere utilizzati per automatizzare l'ottimizzazione il floorplan per ridurre al minimo la lunghezza dei collegamenti, il consumo di energia e altri obiettivi. Questi "floor planners" assistiti da ML automatizzati sono estremamente preziosi man mano che aumenta la complessità dei chip.
- **Simulatori hardware ML:** L'addestramento di modelli di reti neurali profonde per prevedere le prestazioni dei progetti hardware, poiché i simulatori possono accelerare il processo di simulazione di oltre 100 volte rispetto alle simulazioni architettoniche e RTL tradizionali.
- **Traduzione automatica del codice:** La conversione di linguaggi di descrizione hardware come Verilog in implementazioni RTL ottimizzate è fondamentale ma richiede molto tempo. I modelli ML possono essere addestrati per agire come agenti traduttori e automatizzare questo processo.

I vantaggi dell'HDA che utilizza ML sono tempi di progettazione ridotti, ottimizzazioni superiori ed esplorazione di spazi di progettazione troppo complessi per approcci manuali. Ciò può accelerare lo sviluppo hardware e portare a progetti migliori.
Le sfide includono i limiti della generalizzazione ML, la natura black-box di alcune tecniche e compromessi sull'accuratezza. Tuttavia, la ricerca sta rapidamente avanzando per affrontare questi problemi e rendere le soluzioni HDA ML robuste e affidabili per l'uso in produzione. HDA fornisce un'importante via per ML per trasformare la progettazione hardware.

10.9.2 Simulazione e Verifica Hardware Basate su ML

La simulazione e la verifica dei progetti hardware sono fondamentali prima della produzione per garantire che il progetto si comporti come previsto. Gli approcci tradizionali come la simulazione "register-transfer level" (RTL) sono complessi e richiedono molto tempo. Il ML introduce nuove opportunità per migliorare la simulazione e la verifica dell'hardware. Ecco alcuni esempi:

- **Modellazione surrogata per la simulazione:** Modelli surrogati di un progetto altamente accurati possono essere creati utilizzando reti neurali. Questi modelli prevedono gli output dagli input molto più velocemente della simulazione RTL, consentendo una rapida esplorazione dello spazio di progettazione. Aziende come Ansys utilizzano questa tecnica.
- **Simulatori ML:** Grandi modelli di reti neurali possono essere addestrati su simulazioni RTL per imparare a imitare la funzionalità di un progetto hardware. Una volta addestrato, il modello NN può essere un simulatore altamente efficiente per test di regressione e altre attività. Graphcore ha dimostrato un'accelerazione di oltre 100 volte con questo approccio.
- **Verifica formale tramite ML:** La verifica formale dimostra matematicamente le proprietà di un progetto. Le tecniche di ML possono aiutare a generare proprietà di

verifica e imparare a risolvere le complesse prove formali necessarie, automatizzando parti di questo processo impegnativo. Startup come Cortical.io stanno introducendo sul mercato soluzioni di verifica ML formali.

- **Rilevamento di bug:** I modelli ML possono essere addestrati per elaborare progetti hardware e identificare potenziali problemi. Ciò aiuta i progettisti umani a ispezionare progetti complessi e a trovare bug. Facebook ha mostrato modelli di rilevamento di bug per l'hardware dei suoi server.

I principali vantaggi dell'applicazione di ML alla simulazione e alla verifica sono tempi di esecuzione più rapidi per la convalida del progetto, test più rigorosi e riduzione del lavoro umano. Le sfide includono la verifica della correttezza del modello ML e la gestione dei casi limite. ML promette di accelerare significativamente i flussi di lavoro di test.

10.9.3 ML per Architetture Hardware Efficienti

Un obiettivo chiave è la progettazione di architetture hardware ottimizzate per prestazioni, potenza ed efficienza. ML introduce nuove tecniche per automatizzare e migliorare l'esplorazione dello spazio di progettazione dell'architettura per hardware generico e specializzato come gli acceleratori ML. Alcuni esempi promettenti sono:

- **Ricerca di architetture per hardware:** Tecniche di ricerca come algoritmi evolutivi (Kao e Krishna 2020), ottimizzazione bayesiana (Reagen et al. (2017), Bhardwaj et al. (2020)), apprendimento per rinforzo (Kao, Jeong, e Krishna (2020), S. Krishnan et al. (2022)) possono generare automaticamente nuove architetture hardware mutando e mescolando attributi di progettazione come dimensione della cache, numero di unità parallele, larghezza di banda della memoria e così via. Ciò consente un'esplorazione efficiente di ampi spazi di progettazione.
- **Modellazione predittiva per l'ottimizzazione:** I modelli ML possono essere addestrati per prevedere metriche di prestazioni, potenza ed efficienza hardware per una determinata architettura. Questi diventano "modelli surrogati" (S. Krishnan et al. 2023) per una rapida ottimizzazione ed esplorazione dello spazio sostituendo lunghe simulazioni.
- **Ottimizzazione dell'acceleratore specializzato:** Per chip specializzati come unità di elaborazione tensore per AI, tecniche di ricerca architettura automatizzata basate su algoritmi ML (Dan Zhang et al. 2022) promettono di trovare progetti rapidi ed efficienti.

I vantaggi dell'utilizzo di ML includono un'esplorazione dello spazio di progettazione superiore, ottimizzazione automatizzata e riduzione dello sforzo manuale. Le sfide includono lunghi tempi di training per alcune tecniche e limitazioni degli ottimi locali. Tuttavia, ML per l'architettura hardware ha un grande potenziale per rivelare miglioramenti in termini di prestazioni ed efficienza.

10.9.4 ML per Ottimizzare la Produzione e Ridurre i Difetti

Una volta completata la progettazione hardware, si passa alla produzione. Tuttavia, variabilità e difetti durante la produzione possono influire su rese e qualità. Le tecniche ML vengono ora applicate per migliorare i processi di fabbricazione e ridurre i difetti. Ecco alcuni esempi:

- **Manutenzione predittiva:** I modelli ML possono analizzare i dati dei sensori delle apparecchiature nel tempo e identificare segnali che prevedono le esigenze di manutenzione prima del guasto. Ciò consente una manutenzione proattiva, che può essere molto utile nel costoso processo di fabbricazione.
- **Ottimizzazione del processo:** I modelli di apprendimento supervisionato possono essere addestrati sui dati di processo per identificare i fattori che portano a basse

rese. I modelli possono quindi ottimizzare i parametri per migliorare rese, produttività o coerenza.

- **Previsione della resa:** Analizzando i dati di prova da progetti realizzati utilizzando tecniche come alberi di regressione, i modelli ML possono prevedere le rese all'inizio della produzione, consentendo aggiustamenti del processo.
- **Rilevamento dei difetti:** Le tecniche di visione artificiale ML possono essere applicate alle immagini dei progetti per identificare difetti invisibili all'occhio umano. Ciò consente un controllo di qualità di precisione e un'analisi delle cause principali.
- **Analisi proattiva dei guasti:** I modelli ML possono aiutare a prevedere, diagnosticare e prevenire i problemi che portano a difetti e guasti a valle analizzando i dati di processo strutturati e non strutturati.

L'applicazione del ML alla produzione consente l'ottimizzazione dei processi, il controllo di qualità in tempo reale, la manutenzione predittiva e rese più elevate. Le sfide includono la gestione di dati di produzione complessi e varianti. Ma il ML è pronto a trasformare la produzione di semiconduttori.

10.9.5 Verso Modelli di Base per la Progettazione Hardware

Come abbiamo visto, l'apprendimento automatico sta aprendo nuove possibilità nel flusso di lavoro di progettazione hardware, dalle specifiche alla produzione. Tuttavia, le attuali tecniche di ML hanno ancora una portata limitata e richiedono un'ampia progettazione specifica per dominio. La visione a lungo termine è lo sviluppo di sistemi di intelligenza artificiale generali che possono essere applicati con versatilità in tutte le attività di progettazione hardware.

Per realizzare appieno questa visione, sono necessari investimenti e ricerca per sviluppare modelli di base per la progettazione hardware. Si tratta di modelli e architetture ML unificati e generici che possono apprendere complesse competenze di progettazione hardware con i dati di training e gli obiettivi corretti.

La realizzazione di modelli di base per la progettazione hardware end-to-end richiederà quanto segue:

- Accumulare grandi set di dati di alta qualità ed etichettati in tutte le fasi di progettazione hardware per addestrare i modelli di base.
- Progressi nelle tecniche ML multimodali e multi-task per gestire la diversità di dati e attività di progettazione hardware.
- Interfacce e layer di astrazione per collegare i modelli di base ai flussi e agli strumenti di progettazione esistenti.
- Sviluppo di ambienti di simulazione e benchmark per addestrare e testare i modelli di base sulle capacità di progettazione hardware.
- Metodi per spiegare e interpretare le decisioni di progettazione dei modelli ML e le ottimizzazioni per attendibilità e verifica.
- Tecniche di compilazione per ottimizzare i modelli di base per un'implementazione efficiente su piattaforme hardware.

Sebbene siano ancora in corso ricerche significative, i modelli di base rappresentano l'obiettivo a lungo termine più trasformativo per l'infusione dell'IA nel processo della progettazione hardware. Democratizzare la progettazione hardware tramite sistemi ML versatili e automatizzati promette di aprire una nuova era di progettazione di chip ottimizzata, efficiente e innovativa. Il viaggio che ci attende è pieno di sfide e opportunità aperte.

Se sei interessato alla progettazione di architetture per computer assistite da ML (S. Krishnan et al. 2023), invitiamo a leggere Architecture 2.0.

In alternativa, si può guardare Video 10.3 for more details.

Video 10.3: Architecture 2.0

https://www.youtube.com/watch?v=F5Eieaz7u1I&ab_channel=OpenComputeProject

10.10 Conclusione

L'accelerazione hardware specializzata è diventata indispensabile per abilitare applicazioni di intelligenza artificiale performanti ed efficienti, poiché modelli e set di dati esplodono in complessità. Questo capitolo ha esaminato i limiti dei processori generici come le CPU per i carichi di lavoro di intelligenza artificiale. La loro mancanza di parallelismo e di throughput computazionale non consente di addestrare o eseguire rapidamente reti neurali profonde all'avanguardia. Queste motivazioni hanno guidato le innovazioni negli acceleratori personalizzati.

Abbiamo esaminato GPU, TPU, FPGA e ASIC progettati specificamente per le operazioni matematiche intensive inerenti alle reti neurali. Coprendo questo spettro di opzioni, abbiamo mirato a fornire un framework per ragionare attraverso la selezione dell'acceleratore in base a vincoli relativi a flessibilità, prestazioni, potenza, costi e altri fattori.

Abbiamo anche esplorato il ruolo del software nell'abilitazione e nell'ottimizzazione attive dell'accelerazione dell'intelligenza artificiale. Ciò abbraccia astrazioni di programmazione, framework, compilatori e simulatori. Abbiamo discusso della progettazione congiunta hardware-software come metodologia proattiva per la creazione di sistemi di intelligenza artificiale più olistici integrando strettamente l'innovazione degli algoritmi e i progressi hardware.

Ma c'è molto di più in arrivo! Frontiere entusiasmanti come l'informatica analogica, le reti neurali ottiche e l'apprendimento automatico quantistico rappresentano direzioni di ricerca attive che potrebbero sbloccare miglioramenti di ordini di grandezza in termini di efficienza, velocità e scala rispetto ai paradigmi attuali.

In definitiva, l'accelerazione hardware specializzata rimane indispensabile per sbloccare le prestazioni e l'efficienza necessarie per soddisfare la promessa dell'intelligenza artificiale dal cloud all'edge. Ci auguriamo che questo capitolo fornisca utili informazioni di base e approfondimenti sulla rapida innovazione che si sta verificando in questo dominio.

10.11 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

- *Prossimamente.*

Video

- Video 10.1
- Video 10.2
- Video 10.3

🚧 Esercizi

- Esercizio 10.1

⚠ Laboratori

- *Prossimamente.*

11. Benchmarking dell'IA

DALL·E 3 Prompt: Foto di un podio su uno sfondo a tema tecnologico. Su ogni piattaforma del podio ci sono chip AI con design intricati. Il chip in alto ha una medaglia d'oro appesa, il secondo una medaglia d'argento e il terzo una medaglia di bronzo. Sullo sfondo sono ben visibili striscioni con la scritta "AI Olympics".

Il benchmarking è fondamentale per lo sviluppo e la distribuzione di sistemi di machine learning, in particolare applicazioni TinyML. I benchmark consentono agli sviluppatori di misurare e confrontare le prestazioni di diverse architetture di modelli, procedure di training e strategie di distribuzione. Ciò fornisce informazioni chiave su quali approcci funzionano meglio per il problema in questione e sui vincoli dell'ambiente di distribuzione.
Questo capitolo fornirà una panoramica dei benchmark ML più diffusi, le best practice e come utilizzarli per migliorare lo sviluppo del modello e le prestazioni del sistema. Fornire agli sviluppatori gli strumenti e le conoscenze adeguati per effettuare test di benchmark e ottimizzare in modo efficace i propri sistemi, in particolare quelli TinyML.

Obiettivi dell'Apprendimento

- Comprendere lo scopo e gli obiettivi del benchmarking dei sistemi di intelligenza artificiale, tra cui valutazione delle prestazioni, valutazione delle risorse, validazione e altro ancora.
- Scoprire i principali parametri di riferimento, le metriche e le tendenze dei modelli, tra cui accuratezza, equità, complessità, prestazioni ed efficienza energetica.
- Acquisire familiarità con i componenti chiave di un benchmark di intelligenza artificiale, tra cui set di dati, attività, metriche, linee di base, regole di riproducibilità e altro ancora.
- Comprendere la distinzione tra training e inferenza e come ogni fase giustifichi il benchmarking specializzato dei sistemi ML.
- Scoprire i concetti di benchmarking del sistema come produttività, latenza, potenza ed efficienza computazionale.
- Apprezzare l'evoluzione del benchmarking del modello dall'accuratezza a metriche più olistiche come correttezza, robustezza e applicabilità nel mondo reale.
- Riconoscere il ruolo crescente del benchmarking dei dati nella valutazione di problemi come bias, rumore, equilibrio e diversità.
- Comprendere i limiti della valutazione di modelli, dati e sistemi in isolamento e l'esigenza emergente di benchmarking integrato.

11.1 Introduzione

Il benchmarking fornisce le misure essenziali necessarie per guidare il progresso dell'apprendimento automatico e comprendere veramente le prestazioni del sistema. Come ha affermato il fisico Lord Kelvin, "Misurare è conoscere". I benchmark ci consentono di conoscere quantitativamente le capacità di diversi modelli, software e hardware. Consentono agli sviluppatori di ML di misurare il tempo di inferenza, l'utilizzo della memoria, il consumo energetico e altre metriche che caratterizzano un sistema. Inoltre, i benchmark creano processi standardizzati per la misurazione, consentendo confronti equi tra diverse soluzioni.

Quando i benchmark vengono mantenuti nel tempo, diventano fondamentali per catturare i progressi attraverso generazioni di algoritmi, set di dati e hardware. I modelli e le tecniche che stabiliscono nuovi record sui benchmark di ML da un anno all'altro dimostrano miglioramenti tangibili in ciò che è possibile per l'apprendimento automatico "on-device". Utilizzando i benchmark per misurare, i professionisti di ML possono conoscere le capacità reali dei loro sistemi e avere la certezza che ogni passaggio rifletta un progresso autentico verso lo stato dell'arte.

Il benchmarking ha diversi obiettivi e scopi importanti che guidano la sua implementazione per i sistemi di apprendimento automatico.

- **Valutazione delle prestazioni.** Ciò comporta la valutazione di parametri chiave come la velocità, l'accuratezza e l'efficienza di un dato modello. Ad esempio, in un contesto TinyML, è fondamentale confrontare la rapidità con cui un assistente vocale può riconoscere i comandi, poiché ciò valuta le prestazioni in tempo reale.
- **Valutazione della potenza.** Valutare la potenza assorbita da un carico di lavoro insieme alle sue prestazioni equivale alla sua efficienza energetica. Poiché l'impatto ambientale dell'elaborazione ML continua a crescere, il benchmarking dell'energia può consentirci di ottimizzare meglio i sistemi per la sostenibilità.
- **Valutazione delle risorse.** Ciò significa valutare l'impatto del modello sulle risorse critiche del sistema, tra cui durata della batteria, utilizzo della memoria e sovraccarico

computazionale. Un esempio rilevante è il confronto del consumo della batteria di due diversi algoritmi di riconoscimento delle immagini in esecuzione su un dispositivo indossabile.

- **Validazione e verifica.** Il benchmarking aiuta a garantire che il sistema funzioni correttamente e soddisfi i requisiti specificati. Un modo è quello di controllare l'accuratezza di un algoritmo, come un cardiofrequenzimetro su uno smartwatch, rispetto alle letture di apparecchiature di livello medico come forma di validazione clinica.
- **Analisi competitiva.** Ciò consente di confrontare le soluzioni con le offerte concorrenti sul mercato. Ad esempio, il benchmarking di un modello personalizzato di rilevamento di oggetti rispetto ai benchmark TinyML comuni come MobileNet e Tiny-YOLO.
- **Credibilità.** I benchmark accurati sostengono la credibilità delle soluzioni AI e delle organizzazioni che le sviluppano. Dimostrano un impegno verso trasparenza, onestà e qualità, essenziali per creare fiducia con utenti e stakeholder.
- **Regolamentazione e Standardizzazione**. Man mano che il settore dell'AI continua a crescere, cresce anche la necessità di regolamentazione e standardizzazione per garantire che le soluzioni AI siano sicure, etiche ed efficaci. I benchmark accurati e affidabili sono essenziali per questo quadro normativo, poiché forniscono i dati e le prove necessari per valutare la conformità con gli standard del settore e i requisiti legali.

Questo capitolo tratterà i 3 tipi di benchmark AI, le metriche standard, gli strumenti e le tecniche che i progettisti utilizzano per ottimizzare i loro sistemi e le sfide e le tendenze nel benchmarking.

11.2 Contesto Storico

11.2.1 Benchmark delle Prestazioni

L'evoluzione dei benchmark nell'informatica illustra vividamente l'incessante ricerca dell'eccellenza e dell'innovazione da parte del settore. Nei primi giorni dell'informatica, negli anni '60 e '70, i benchmark erano rudimentali e progettati per i mainframe. Ad esempio, il benchmark Whetstone, che prende il nome dal compilatore Whetstone ALGOL, è stato uno dei primi test standardizzati per misurare le prestazioni aritmetiche in virgola mobile di una CPU. Questi benchmark pionieristici hanno spinto i produttori a perfezionare le loro architetture e algoritmi per ottenere punteggi di benchmark migliori.

Gli anni '80 hanno segnato un cambiamento significativo con l'ascesa dei personal computer. Mentre aziende come IBM, Apple e Commodore gareggiavano per quote di mercato, i benchmark sono diventati strumenti essenziali per consentire una concorrenza leale. I benchmark CPU SPEC, introdotti dalla System Performance Evaluation Cooperative (SPEC), hanno stabilito test standardizzati che consentono confronti oggettivi tra diverse macchine. Questa standardizzazione ha creato un ambiente competitivo, spingendo i produttori di chip e i creatori di sistemi a migliorare continuamente le loro offerte hardware e software.

Gli anni '90 hanno portato l'era delle applicazioni e dei videogiochi "graphics-intensive". La necessità di benchmark per valutare le prestazioni delle schede grafiche ha portato alla creazione di 3DMark da parte di Futuremark. Mentre i giocatori e i professionisti cercavano schede grafiche ad alte prestazioni, aziende come NVIDIA e AMD sono state spinte a una rapida innovazione, portando a importanti progressi nella tecnologia GPU come gli shader programmabili.

Gli anni 2000 hanno visto un'impennata di telefoni cellulari e dispositivi portatili come i tablet. Con la portabilità è arrivata la sfida di bilanciare prestazioni e consumo energetico. Benchmark come MobileMark di BAPCo hanno valutato velocità e durata della batteria. Ciò ha spinto le aziende a sviluppare System-on-Chip (SOC) più efficienti dal punto di vista energetico, portando all'emergere di architetture come ARM che hanno dato priorità all'efficienza energetica.
L'attenzione dell'ultimo decennio si è spostata verso il cloud computing, i big data e l'intelligenza artificiale. I provider di servizi cloud come Amazon Web Services e Google Cloud competono su prestazioni, scalabilità e convenienza. I benchmark specifici del cloud come CloudSuite sono diventati essenziali, spingendo i provider a ottimizzare la propria infrastruttura per servizi migliori.

11.2.2 Benchmark Energetici

Il consumo energetico e le preoccupazioni ambientali hanno acquisito importanza negli ultimi anni, rendendo il benchmarking energetico sempre più importante nel settore. Questo cambiamento è iniziato a metà degli anni 2000, quando i processori e i sistemi hanno iniziato a raggiungere i limiti di raffreddamento e la scalabilità è diventata un aspetto cruciale della costruzione di sistemi su larga scala grazie ai progressi di Internet. Da allora, le considerazioni energetiche si sono espanse fino a comprendere tutte le aree dell'informatica, dai dispositivi personali ai data center su larga scala.
Il benchmarking energetico mira a misurare l'efficienza energetica dei sistemi informatici, valutando le prestazioni in relazione al consumo energetico. Ciò è fondamentale per diversi motivi:

- **Impatto ambientale:** Con la crescente impronta di carbonio del settore tecnologico, c'è un'urgente necessità di ridurre il consumo energetico.
- **Costi operativi:** Le spese energetiche costituiscono una parte significativa dei costi operativi del data center.
- **Longevità del dispositivo:** Per i dispositivi mobili, l'efficienza energetica ha un impatto diretto sulla durata della batteria e sull'esperienza utente.

In questo ambito sono emersi diversi benchmark chiave:

- **SPEC Power:** Introdotto nel 2007, SPEC Power è stato uno dei primi benchmark standard del settore per la valutazione delle caratteristiche di potenza e prestazioni dei server.
- **Green500:** L'elenco Green500 classifica i supercomputer in base all'efficienza energetica, integrando l'elenco TOP500 incentrato sulle prestazioni.
- **Energy Star:** Pur non essendo un benchmark in sé, il programma di certificazione ENERGY STAR for Computers ha spinto i produttori a migliorare l'efficienza energetica dell'elettronica di consumo.

Il benchmarking energetico affronta sfide uniche, come la contabilizzazione di diversi carichi di lavoro e configurazioni di sistema e la misura accurata del consumo energetico su una gamma di hardware che varia da microWatt a megawatt nel consumo energetico. Man mano che l'IA e l'edge computing continuano a crescere, è probabile che il benchmarking energetico diventi ancora più critico, guidando lo sviluppo di ottimizzazioni hardware e software AI specializzate ed efficienti dal punto di vista energetico.

11.2.3 Benchmark Personalizzati

Oltre ai benchmark standard del settore, ci sono benchmark personalizzati specificamente progettati per soddisfare i requisiti unici di una particolare applicazione o attività. Sono personalizzati in base alle esigenze specifiche dell'utente o dello sviluppatore, assicurando

che le metriche delle prestazioni siano direttamente pertinenti all'uso previsto del modello o del sistema di intelligenza artificiale. I benchmark personalizzati possono essere creati da singole organizzazioni, ricercatori o sviluppatori e sono spesso utilizzati insieme ai benchmark standard del settore per fornire una valutazione completa delle prestazioni dell'intelligenza artificiale.

Ad esempio, un ospedale potrebbe sviluppare un benchmark per valutare un modello di intelligenza artificiale per prevedere la riammissione dei pazienti. Questo benchmark incorporerebbe metriche pertinenti alla popolazione di pazienti dell'ospedale, come dati demografici, anamnesi e fattori sociali. Allo stesso modo, il benchmark di rilevamento delle frodi di un istituto finanziario potrebbe concentrarsi sull'identificazione accurata delle transazioni fraudolente riducendo al minimo i falsi positivi. Nel settore automobilistico, un benchmark di veicoli autonomi potrebbe dare priorità alle prestazioni in diverse condizioni, alla risposta agli ostacoli e alla sicurezza. I rivenditori potrebbero confrontare i sistemi di raccomandazione utilizzando il tasso di clic, il tasso di conversione e la soddisfazione del cliente. Le aziende manifatturiere potrebbero confrontare i sistemi di controllo qualità in base all'identificazione dei difetti, all'efficienza e alla riduzione degli sprechi. In ogni settore, i benchmark personalizzati forniscono alle organizzazioni criteri di valutazione su misura per le loro esigenze e il loro contesto unici. Ciò consente una valutazione più significativa di quanto i sistemi di intelligenza artificiale soddisfino i requisiti.

Il vantaggio dei benchmark personalizzati risiede nella loro flessibilità e pertinenza. Possono essere progettati per testare aspetti specifici delle prestazioni critici per il successo della soluzione di intelligenza artificiale nella sua applicazione prevista. Ciò consente una valutazione più mirata e accurata delle capacità del modello o del sistema di intelligenza artificiale. I benchmark personalizzati forniscono anche informazioni preziose sulle prestazioni delle soluzioni di intelligenza artificiale in scenari reali, il che può essere cruciale per identificare potenziali problemi e aree di miglioramento.

Nell'intelligenza artificiale, i benchmark svolgono un ruolo cruciale nel guidare il progresso e l'innovazione. Sebbene i benchmark siano stati a lungo utilizzati nell'informatica, la loro applicazione all'apprendimento automatico è relativamente recente. I benchmark incentrati sull'intelligenza artificiale forniscono metriche standardizzate per valutare e confrontare le prestazioni di diversi algoritmi, architetture di modelli e piattaforme hardware.

11.2.4 Consenso della Comunità

Una prerogativa fondamentale affinché un benchmark abbia un impatto è che deve riflettere le priorità e i valori condivisi della più ampia comunità di ricerca. I benchmark progettati in modo isolato rischiano di non ottenere accettazione se trascurano metriche chiave considerate importanti dai gruppi leader. Attraverso uno sviluppo collaborativo con la partecipazione aperta di laboratori accademici, aziende e altri stakeholder, i benchmark possono incorporare un contributo collettivo su capacità critiche che vale la pena misurare. Ciò aiuta a garantire che i benchmark valutino aspetti che la comunità concorda siano essenziali per far progredire il campo. Il processo di raggiungimento dell'allineamento su attività e metriche supporta di per sé la convergenza su ciò che conta di più.

Inoltre, i benchmark pubblicati con ampia co-paternità da istituzioni rispettate hanno autorità e validità che convincono la comunità ad adottarli come standard affidabili. I benchmark percepiti come distorti da particolari interessi aziendali o istituzionali generano scetticismo. Anche il coinvolgimento continuo della comunità attraverso workshop e sfide è fondamentale dopo la versione iniziale, ed è ciò che, ad esempio, ha portato al successo di ImageNet. Col progredire della ricerca, la partecipazione collettiva consente un continuo perfezionamento ed espansione dei benchmark nel tempo.

Infine, i benchmark sviluppati dalla comunità rilasciati con accesso aperto accelerano l'adozione e l'implementazione coerente. Abbiamo condiviso codice open source, documentazione, modelli e infrastrutture per ridurre le barriere che impediscono ai gruppi di confrontare le soluzioni su un piano di parità utilizzando implementazioni standardizzate. Questa coerenza è fondamentale per confronti equi. Senza coordinamento, laboratori e aziende potrebbero implementare i benchmark in modo diverso, riducendo la riproducibilità dei risultati.
Il consenso della comunità conferisce ai benchmark una rilevanza duratura, mentre la frammentazione confonde. Attraverso lo sviluppo collaborativo e un funzionamento trasparente, i benchmark possono diventare standard autorevoli per monitorare i progressi. Molti dei benchmark di cui parliamo in questo capitolo sono stati sviluppati e creati dalla comunità, per la comunità, ed è questo che alla fine ha portato al loro successo.

11.3 Benchmark AI: Sistema, Modello e Dati

La necessità di un benchmarking completo diventa fondamentale man mano che i sistemi AI diventano più complessi e onnipresenti. In questo contesto, i benchmark sono spesso classificati in tre categorie principali: Hardware, Modello e Dati. Analizziamo perché ognuno di questi gruppi è essenziale e il significato della valutazione dell'AI da queste tre dimensioni distinte:

11.3.1 Benchmark di Sistema

I calcoli AI, in particolare quelli nel deep learning, richiedono molte risorse. L'hardware su cui vengono eseguiti questi calcoli svolge un ruolo importante nel determinare la velocità, l'efficienza e la scalabilità delle soluzioni AI. Di conseguenza, i benchmark hardware aiutano a valutare le prestazioni di CPU, GPU, TPU e altri acceleratori nelle attività AI. Comprendendone le prestazioni, gli sviluppatori possono scegliere quali piattaforme hardware si adattano meglio a specifiche applicazioni AI. Inoltre, i produttori di hardware utilizzano questi benchmark per identificare aree di miglioramento, guidando l'innovazione nei progetti di chip specifici per AI.

11.3.2 Benchmark del Modello

L'architettura, le dimensioni e la complessità dei modelli AI variano notevolmente. Modelli diversi hanno diverse esigenze di calcolo e offrono diversi livelli di accuratezza ed efficienza. I benchmark dei modelli aiutano a valutare le prestazioni di varie architetture AI su attività standardizzate. Forniscono informazioni sulla velocità, l'accuratezza e le richieste di risorse di diversi modelli. Eseguendo il benchmarking dei modelli, i ricercatori possono identificare le architetture più performanti per attività specifiche, guidando la comunità AI verso soluzioni più efficienti ed efficaci. Inoltre, questi benchmark aiutano a monitorare i progressi della ricerca sull'intelligenza artificiale, mostrando i progressi nella progettazione e nell'ottimizzazione dei modelli.

11.3.3 Benchmark dei Dati

L'intelligenza artificiale, in particolare l'apprendimento automatico, è intrinsecamente basata sui dati. La qualità, le dimensioni e la diversità dei dati influenzano l'efficacia dell'addestramento e la capacità di generalizzazione dei modelli di intelligenza artificiale. I benchmark dei dati si concentrano sui set di dati utilizzati nell'addestramento e nella valutazione dell'intelligenza artificiale. Forniscono set di dati standardizzati che la comunità può utilizzare per addestrare e testare i modelli, garantendo parità di condizioni per i confronti. Inoltre, questi benchmark evidenziano le sfide relative alla qualità, alla diversità e alla rappresentazione dei dati, spingendo la comunità ad affrontare bias e lacune nei dati di addestramento dell'intelligenza artificiale. Comprendendo i benchmark dei dati, i ricercatori

possono anche valutare come i modelli potrebbero comportarsi in scenari reali, garantendo robustezza e affidabilità.
Nelle restanti sezioni, discuteremo ciascuno di questi tipi di benchmark. L'attenzione sarà rivolta a un'esplorazione approfondita dei benchmark di sistema, poiché sono fondamentali per comprendere e migliorare le prestazioni del sistema di apprendimento automatico. Parleremo brevemente dei benchmark dei modelli e dei dati per una prospettiva completa, ma l'enfasi e la maggior parte del contenuto saranno dedicati ai benchmark di sistema.

11.4 Benchmarking di Sistema

11.4.1 Granularità

Il benchmarking del sistema di apprendimento automatico fornisce un approccio strutturato e sistematico per valutare le prestazioni di un sistema in diverse dimensioni. Data la complessità dei sistemi ML, possiamo analizzare le loro prestazioni attraverso diversi livelli di granularità e ottenere una visione completa dell'efficienza del sistema, identificare potenziali colli di bottiglia e individuare le aree di miglioramento. A tal fine, nel corso degli anni si sono evoluti vari tipi di benchmark che continuano a persistere.
Figura 11.1 illustra i diversi livelli di granularità di un sistema ML. A livello di applicazione, i benchmark end-to-end valutano le prestazioni complessive del sistema, considerando fattori come la pre-elaborazione dei dati, l'addestramento del modello e l'inferenza. Mentre a livello di modello, i benchmark si concentrano sulla valutazione dell'efficienza e dell'accuratezza di modelli specifici. Ciò include la valutazione di quanto bene i modelli si generalizzano a nuovi dati e della loro efficienza computazionale durante l'addestramento e l'inferenza. Inoltre, il benchmarking può estendersi all'infrastruttura hardware e software, esaminando le prestazioni di singoli componenti come GPU o TPU.

Figura 11.1: Granularità del sistema ML.

11.4.1.1 Micro Benchmark

I micro-benchmark nell'IA sono specializzati e valutano componenti distinti o operazioni specifiche all'interno di un processo di apprendimento automatico più ampio. Questi benchmark si concentrano su singole attività, offrendo approfondimenti sulle richieste computazionali di un particolare layer di rete neurale, l'efficienza di un'unica tecnica di ottimizzazione o la produttività di una specifica funzione di attivazione. Ad esempio, i professionisti potrebbero utilizzare i micro-benchmark per misurare il tempo di calcolo richiesto da un layer convoluzionale in un modello di deep learning o per valutare la velocità di preelaborazione che alimenta i dati nel modello. Tali valutazioni granulari sono fondamentali per la messa a punto e l'ottimizzazione di aspetti discreti dei modelli di IA, assicurando che ogni componente funzioni al massimo del suo potenziale.
Questi tipi di microbenchmark includono lo zoom su operazioni o componenti molto specifiche della pipeline AI, come le seguenti:

- **Operazioni Tensoriali:** Librerie come cuDNN (di NVIDIA) spesso hanno benchmark per misurare le prestazioni di singole operazioni tensoriali, come convoluzioni o moltiplicazioni di matrici, che sono fondamentali per i calcoli del deep learning.
- **Funzioni di Attivazione:** Benchmark che misurano la velocità e l'efficienza di varie funzioni di attivazione come ReLU, Sigmoid o Tanh in isolamento.
- **Benchmark di Layer:** Valutazioni dell'efficienza computazionale di distinti layer di rete neurale, come blocchi LSTM o Transformer, quando si opera su dimensioni di input standardizzate.

Esempio: DeepBench, introdotto da Baidu, è un buon esempio di qualcosa che valuta quanto sopra. DeepBench valuta le prestazioni delle operazioni di base nei modelli di deep learning, fornendo informazioni su come diverse piattaforme hardware gestiscono l'addestramento e l'inferenza delle reti neurali.

Esercizio 11.1: Benchmarking di Sistema - Operazioni Tensoriali

Ci si è mai chiesto come mai i filtri immagine diventano così veloci? Librerie speciali come cuDNN potenziano quei calcoli su determinati hardware. In questo Colab, useremo cuDNN con PyTorch per velocizzare il filtraggio delle immagini. Lo si consideri un piccolo benchmark, che mostra come il software giusto può sbloccare la potenza della GPU!

Open in Colab

11.4.1.2 Macro Benchmark

I macro benchmark forniscono una visione olistica, valutando le prestazioni end-to-end di interi modelli di apprendimento automatico o sistemi di intelligenza artificiale completi. Invece di concentrarsi sulle singole operazioni, i macro benchmark valutano l'efficacia collettiva dei modelli in scenari o attività del mondo reale. Ad esempio, un macro benchmark potrebbe valutare le prestazioni complete di un modello di apprendimento profondo che esegue la classificazione delle immagini su un set di dati come ImageNet. Ciò include la misura dell'accuratezza, della velocità di calcolo e del consumo di risorse. Allo stesso modo, si potrebbero misurare il tempo e le risorse cumulativi necessari per addestrare un modello di elaborazione del linguaggio naturale su corpora di testo estesi o valutare le prestazioni di un intero sistema di raccomandazione, dall'inserimento dei dati agli output finali specifici dell'utente.

Esempi: Questi benchmark valutano il modello di intelligenza artificiale:

- MLPerf Inference (V. J. Reddi et al. 2020): Un set di benchmark standard per misurare le prestazioni di software e hardware di apprendimento automatico. MLPerf ha una suite di benchmark dedicati per scale specifiche, come MLPerf Mobile per dispositivi di classe mobile e MLPerf Tiny, che si concentra su microcontrollori e altri dispositivi con risorse limitate.
- MLMark di EEMBC: Una suite di benchmarking per valutare le prestazioni e l'efficienza energetica dei dispositivi embedded che eseguono carichi di lavoro di apprendimento automatico. Questo benchmark fornisce informazioni su come diverse piattaforme hardware gestiscono attività come il riconoscimento delle immagini o l'elaborazione audio.
- AI-Benchmark (Ignatov et al. 2018): Uno strumento di benchmarking progettato per dispositivi Android, valuta le prestazioni delle attività di intelligenza artificiale sui dispositivi mobili, comprendendo vari scenari del mondo reale come il riconoscimento delle immagini, l'analisi dei volti e il riconoscimento ottico dei caratteri.

11.4.1.3 Benchmark end-to-end

I benchmark end-to-end forniscono una valutazione completa che si estende oltre i confini del modello di IA stesso. Invece di concentrarsi esclusivamente sull'efficienza o l'accuratezza computazionale di un modello di apprendimento automatico, questi benchmark comprendono l'intera pipeline di un sistema di IA. Ciò include la pre-elaborazione iniziale dei dati, le prestazioni del modello principale, la post-elaborazione degli output del modello e altri componenti integrali come l'archiviazione e le interazioni di rete.

La pre-elaborazione dei dati è la prima fase in molti sistemi di IA, trasformando i dati grezzi in un formato adatto per l'addestramento o l'inferenza del modello. L'efficienza, la scalabilità e l'accuratezza di queste fasi di pre-elaborazione sono vitali per le prestazioni complessive del sistema. I benchmark end-to-end valutano questa fase, assicurando che la pulizia dei dati, la normalizzazione, l'aumento o qualsiasi altro processo di trasformazione non diventi un collo di bottiglia.

Anche la fase di post-elaborazione è al centro dell'attenzione. Ciò comporta l'interpretazione degli output grezzi del modello, eventualmente la conversione dei punteggi in categorie significative, il filtraggio dei risultati o persino l'integrazione con altri sistemi. Nelle applicazioni del mondo reale, questa fase è fondamentale per fornire informazioni fruibili e i benchmark end-to-end ne garantiscono l'efficienza e l'efficacia.

Oltre alle operazioni di base dell'IA, altri componenti del sistema sono importanti per le prestazioni complessive e l'esperienza utente. Le soluzioni di archiviazione, basate su cloud, on-premise o ibride, possono avere un impatto significativo sui tempi di recupero e archiviazione dei dati, in particolare con vasti set di dati di IA. Allo stesso modo, le interazioni di rete, vitali per le soluzioni di IA basate su cloud o per i sistemi distribuiti, possono diventare colli di bottiglia delle prestazioni se non ottimizzate. I benchmark end-to-end valutano in modo olistico questi componenti, assicurando che l'intero sistema funzioni senza problemi, dal recupero dei dati alla consegna dell'output finale.

Ad oggi, non esistono benchmark end-to-end pubblici che tengano conto del ruolo dell'archiviazione dei dati, della rete e delle prestazioni di elaborazione. Si può sostenere che MLPerf Training and Inference si avvicini all'idea di un benchmark end-to-end, ma si concentrano esclusivamente sulle prestazioni del modello ML e non rappresentano scenari di distribuzione nel mondo reale di come i modelli vengono utilizzati sul campo. Tuttavia, forniscono un segnale molto utile che aiuta a valutare le prestazioni del sistema AI.

Data la specificità intrinseca del benchmarking end-to-end, viene in genere eseguito internamente in un'azienda "strumentando" [inserendo punti di controllo] distribuzioni di produzione reali di AI. Ciò consente agli ingegneri di avere una comprensione e una ripartizione realistiche delle prestazioni, ma data la sensibilità e la specificità delle informazioni, raramente vengono segnalate all'esterno dell'azienda.

11.4.1.4 Comprendere i Compromessi

Diversi problemi sorgono nelle diverse fasi di un sistema di intelligenza artificiale. I micro-benchmark aiutano a mettere a punto i singoli componenti, i macro-benchmark aiutano a perfezionare le architetture o gli algoritmi del modello e i benchmark end-to-end guidano l'ottimizzazione dell'intero flusso di lavoro. Comprendendo dove si trova un problema, gli sviluppatori possono applicare ottimizzazioni mirate.

Inoltre, mentre i singoli componenti di un sistema di intelligenza artificiale potrebbero funzionare in modo ottimale in isolamento, possono emergere colli di bottiglia quando interagiscono. I benchmark end-to-end, in particolare, sono fondamentali per garantire che l'intero sistema, quando funziona collettivamente, soddisfi gli standard di prestazioni ed efficienza desiderati.

Infine, le organizzazioni possono prendere decisioni informate su dove allocare le risorse individuando colli di bottiglia o inefficienze nelle prestazioni. Ad esempio, se i micro-benchmark rivelano inefficienze in specifiche operazioni tensoriali, gli investimenti possono essere indirizzati verso acceleratori hardware specializzati. Al contrario, se i benchmark end-to-end indicano problemi di recupero dei dati, gli investimenti potrebbero essere incanalati verso soluzioni di archiviazione migliori.

11.4.2 Componenti dei Benchmark

In sostanza, un benchmark AI è più di un semplice test o punteggio; è un framework di valutazione completo. Per comprenderlo in modo approfondito, analizziamo i componenti tipici che compongono un benchmark AI.

11.4.2.1 Dataset Standardizzati

I set di dati fungono da base per la maggior parte dei benchmark AI. Forniscono un set di dati coerente su cui i modelli vengono addestrati e valutati, garantendo parità di condizioni per i confronti.
Esempio: ImageNet, un set di dati su larga scala contenente milioni di immagini etichettate che abbracciano migliaia di categorie, è uno standard di benchmarking popolare per le attività di classificazione delle immagini.

11.4.2.2 Attività Predefinite

Un benchmark dovrebbe avere un obiettivo o un compito chiaro che i modelli mirano a raggiungere. Questo compito definisce il problema che il sistema AI sta cercando di risolvere.
Esempio: I compiti per i benchmark di elaborazione del linguaggio naturale potrebbero includere analisi del “sentiment”, riconoscimento di entità denominate o traduzione automatica.

11.4.2.3 Metriche di Valutazione

Una volta definito un task, i benchmark richiedono parametri per quantificare le prestazioni. Questi parametri offrono misure oggettive per confrontare diversi modelli o sistemi. Nei task di classificazione, parametri come accuratezza, precisione, richiamo e punteggio F1 sono comunemente utilizzati. Errori quadratici medi o assoluti potrebbero essere utilizzati per i task di regressione. Possiamo anche misurare la potenza consumata dall’esecuzione del benchmark per calcolare l’efficienza energetica.

11.4.2.4 Baseline e Modelli Baseline

I benchmark spesso includono modelli “baseline” o implementazioni di riferimento. Di solito servono come punti di partenza o standard minimi di prestazione per confrontare nuovi modelli o nuove tecniche. I modelli “baseline” aiutano i ricercatori a misurare l’efficacia di nuovi algoritmi.
Nelle suite di benchmark, modelli semplici come la regressione lineare o le reti neurali di base sono spesso le baseline comuni. Queste forniscono un contesto quando si valutano modelli più complessi. Confrontando questi modelli più semplici, i ricercatori possono quantificare i miglioramenti derivanti da approcci avanzati.
Le metriche delle prestazioni variano in base all’attività, ma ecco alcuni esempi:

- Le attività di classificazione utilizzano metriche come accuratezza, precisione, richiamo e punteggio F1.
- Le attività di regressione utilizzano spesso l’errore quadratico medio o l’errore assoluto medio.

11.4.2.5 Specifiche Hardware e Software

Data la variabilità introdotta da diverse configurazioni hardware e software, i benchmark spesso specificano o documentano gli ambienti hardware e software in cui vengono condotti i test.
Esempio: Un benchmark AI potrebbe indicare che le valutazioni sono state condotte su una GPU NVIDIA Tesla V100 utilizzando TensorFlow v2.4.

11.4.2.6 Condizioni Ambientali

Poiché fattori esterni possono influenzare i risultati del benchmark, è essenziale controllare o documentare condizioni come temperatura, fonte di alimentazione o processi di background del sistema.
Esempio: I benchmark AI mobili potrebbero specificare che i test sono stati condotti a temperatura ambiente con dispositivi collegati a una fonte di alimentazione per eliminare le variazioni del livello della batteria.

11.4.2.7 Regole di Riproducibilità

Per garantire che i benchmark siano credibili e possano essere replicati da altri nella comunità, spesso includono protocolli dettagliati che coprono tutto, dai "random seed" utilizzati agli iperparametri esatti.
Esempio: Un benchmark per un'attività di learning di rinforzo potrebbe specificare gli episodi esatti dell'addestramento, i rapporti di esplorazione-sfruttamento e le strutture di ricompensa utilizzate.

11.4.2.8 Linee Guida per l'Interpretazione dei Risultati

Oltre ai punteggi o alle metriche pure, i benchmark spesso forniscono linee guida o contesto per interpretare i risultati, aiutando i professionisti a comprendere le implicazioni più ampie.
Esempio: Un benchmark potrebbe evidenziare che, sebbene il Modello A abbia ottenuto un punteggio più alto del Modello B in termini di accuratezza, offre migliori prestazioni in tempo reale, rendendolo più adatto per applicazioni sensibili al fattore tempo.

11.4.3 Training vs. inferenza

Il ciclo di vita dello sviluppo di un modello di apprendimento automatico prevede due fasi critiche: addestramento e inferenza. Training, come forse ricorderete, è il processo di apprendimento di pattern dai dati per creare il modello. L'inferenza si riferisce al modello che fa previsioni su nuovi dati non etichettati. Entrambe le fasi svolgono ruoli indispensabili ma distinti. Di conseguenza, ogni fase garantisce un rigoroso benchmarking per valutare metriche delle prestazioni come velocità, accuratezza ed efficienza computazionale.
Il benchmarking della fase di training fornisce informazioni su come diverse architetture di modelli, valori di iperparametri e algoritmi di ottimizzazione influiscono sul tempo e sulle risorse necessarie per il training del modello. Ad esempio, il benchmarking mostra come la profondità della rete neurale influisce sul tempo di training su un dato set di dati. Il benchmarking rivela anche come acceleratori hardware come GPU e TPU possono accelerare il training.
D'altro canto, il benchmark dell'inferenza valuta le prestazioni del modello in condizioni reali dopo la distribuzione. Le metriche chiave includono latenza, throughput, footprint di memoria e consumo energetico. Questo tipo di benchmarking determina se un modello soddisfa i requisiti della sua applicazione target in termini di tempo di risposta e vincoli del dispositivo. Tuttavia, ne discuteremo ampiamente per garantire una comprensione generale.

11.4.4 I Benchmark del Training

Il training [addestramento] rappresenta la fase in cui il sistema elabora e assimila dati grezzi per adattare e perfezionare i propri parametri. Pertanto, è un'attività algoritmica e comporta

considerazioni a livello di sistema, tra cui pipeline di dati, archiviazione, risorse di elaborazione e meccanismi di orchestrazione. L'obiettivo è garantire che il sistema ML possa apprendere in modo efficiente dai dati, ottimizzando sia le prestazioni del modello sia l'utilizzo delle risorse del sistema.

11.4.4.1 Scopo

Dal punto di vista dei sistemi ML, i benchmark del training valutano quanto bene il sistema si adatta all'aumento dei volumi di dati e delle richieste di elaborazione. Si tratta di comprendere l'interazione tra hardware, software e pipeline di dati nel processo di training.
Si consideri un sistema ML distribuito progettato per il training su vasti set di dati, come quelli utilizzati nelle raccomandazioni di prodotti di e-commerce su larga scala. Un benchmark di training valuterebbe l'efficienza con cui il sistema si adatta a più nodi, gestisce lo sharding [partizionamento] dei dati e gestisce guasti o abbandoni dei nodi durante il training.
I benchmark di training valutano l'utilizzo di CPU, GPU, memoria e rete durante la fase di training, guidando le ottimizzazioni del sistema. Quando si addestra un modello in un sistema ML basato su cloud, è fondamentale capire come vengono utilizzate le risorse. Le GPU vengono sfruttate appieno? C'è un sovraccarico di memoria non necessario? I benchmark possono evidenziare colli di bottiglia o inefficienze nell'utilizzo delle risorse, con conseguenti risparmi sui costi e miglioramenti delle prestazioni.
Il training di un modello ML è subordinato alla consegna tempestiva ed efficiente dei dati. I benchmark in questo contesto valuterebbero anche l'efficienza delle pipeline di dati, la velocità di preelaborazione dei dati e i tempi di recupero dell'archiviazione. Per i sistemi di analisi in tempo reale, come quelli utilizzati nel rilevamento delle frodi, la velocità con cui i dati di training vengono ingeriti, preelaborati e immessi nel modello può essere critica. I benchmark valuterebbero la latenza delle pipeline di dati, l'efficienza dei sistemi di archiviazione (come SSD rispetto a HDD) e la velocità delle attività di aumento o trasformazione dei dati.

11.4.4.2 Metriche

Se viste da una prospettiva di sistema, le metriche di training offrono informazioni che trascendono gli indicatori di prestazioni algoritmiche convenzionali. Queste metriche misurano l'efficacia di apprendimento del modello e misurano l'efficienza, la scalabilità e la robustezza dell'intero sistema ML durante la fase di training. Analizziamo più a fondo queste metriche e il loro significato.
Le seguenti metriche sono spesso considerate importanti:

1. **Tempo di training:** Il tempo necessario per addestrare un modello da zero fino a raggiungere un livello di prestazioni soddisfacente. Misura direttamente le risorse di elaborazione necessarie per addestrare un modello. Ad esempio, il BERT di Google (Devlin et al. 2019) è un modello di elaborazione del linguaggio naturale che richiede diversi giorni per l'addestramento su un corpus enorme di dati di testo utilizzando più GPU. Il lungo tempo di training è una sfida significativa in termini di consumo di risorse e costi. In alcuni casi, i benchmark possono invece misurare la produttività del training (campioni di training per unità di tempo). La produttività può essere calcolata molto più velocemente e facilmente del tempo di addestramento, ma potrebbe oscurare le metriche che ci interessano davvero (ad esempio, il tempo di addestramento).
2. **Scalabilità:** Quanto bene il processo di addestramento può gestire gli aumenti delle dimensioni dei dati o della complessità del modello. La scalabilità può essere valutata misurando il tempo di addestramento, l'utilizzo della memoria e altri consumi di risorse all'aumentare delle dimensioni dei dati o della complessità del modello. Il modello GPT-3 di OpenAI (Brown et al. 2020) ha 175 miliardi di parametri, il che lo rende uno dei più grandi modelli linguistici esistenti. L'addestramento di GPT-3 ha

richiesto notevoli sforzi ingegneristici per scalare il processo di addestramento in modo da gestire le enormi dimensioni del modello. Ciò ha comportato l'utilizzo di hardware specializzato, addestramento distribuito e altre tecniche per garantire che il modello potesse essere addestrato in modo efficiente.

3. **Utilizzo delle Risorse:** La misura in cui il processo di addestramento utilizza le risorse di calcolo disponibili come CPU, GPU, memoria e I/O del disco. Un elevato utilizzo delle risorse può indicare un processo di training efficiente, mentre un basso utilizzo può suggerire colli di bottiglia o inefficienze. Ad esempio, il training di una rete neurale convoluzionale (CNN) per la classificazione delle immagini richiede notevoli risorse GPU. L'utilizzo di configurazioni multi-GPU e l'ottimizzazione del codice di training per l'accelerazione GPU possono migliorare notevolmente l'utilizzo delle risorse e l'efficienza del training.
4. **Consumo di Memoria:** La quantità di memoria utilizzata dal processo di training. Il consumo di memoria può essere un fattore limitante per il training di modelli o set di dati di grandi dimensioni. Ad esempio, i ricercatori di Google hanno dovuto affrontare notevoli sfide di consumo di memoria durante il training di BERT. Il modello ha centinaia di milioni di parametri, che richiedono grandi quantità di memoria. I ricercatori hanno dovuto sviluppare tecniche per ridurre il consumo di memoria, come il checkpointing del gradiente e il parallelismo del modello.
5. **Consumo Energetico:** L'energia consumata durante il training. Man mano che i modelli di apprendimento automatico diventano più complessi, il consumo energetico è diventato un fattore importante da considerare. Il training di grandi modelli di apprendimento automatico può consumare molta energia, e quindi molto carbonio. Ad esempio, si è stimato che l'addestramento di GPT-3 di OpenAI abbia un'impronta di carbonio equivalente a un viaggio in auto di 700.000 chilometri.
6. **Throughput:** l numero di campioni di addestramento elaborati per unità di tempo. Un throughput [produttività] più elevato indica generalmente un processo di addestramento più efficiente. La produttività è una metrica importante da considerare quando si addestra un sistema di raccomandazione per una piattaforma di e-commerce. Una produttività elevata assicura che il modello possa elaborare rapidamente grandi volumi di dati di interazione dell'utente, il che è fondamentale per mantenere la pertinenza e l'accuratezza delle raccomandazioni. Ma è anche importante capire come bilanciare la produttività con i limiti di latenza. Pertanto, un vincolo di produttività limitato dalla latenza viene spesso imposto agli accordi sul livello di servizio per le distribuzioni di applicazioni del data center.
7. **Costo:** Il costo della training di un modello può includere sia risorse computazionali che umane. Il costo è importante quando si considera la praticità e la fattibilità del training di modelli grandi o complessi. Si stima che l'addestramento di modelli di linguaggio grandi come GPT-3 costi milioni di dollari. Questo costo include risorse computazionali, elettriche e umane necessarie per lo sviluppo e l'addestramento del modello.
8. **Tolleranza agli Errori e Robustezza:** La capacità del processo di training di gestire guasti o errori senza bloccarsi o produrre risultati errati. Questo è importante per garantire l'affidabilità del processo di addestramento. Errori di rete o malfunzionamenti hardware possono verificarsi in uno scenario reale in cui un modello di apprendimento automatico viene addestrato su un sistema distribuito. Negli ultimi anni, è diventato abbondantemente chiaro che gli errori derivanti dalla corruzione "silenziosa" dei dati sono emersi come un problema importante. Un processo di addestramento affidabile e tollerante agli errori può recuperare da tali errori senza compromettere l'integrità del modello.

9. **Facilità d'Uso e Flessibilità:** La facilità con cui il processo di addestramento può essere impostato e utilizzato e la sua flessibilità nella gestione di diversi tipi di dati e modelli. In aziende come Google, l'efficienza può talvolta essere misurata dal numero di anni di "Software Engineer (SWE)" risparmiati poiché ciò si traduce direttamente in impatto. La facilità d'uso e la flessibilità possono ridurre il tempo e lo sforzo necessari per addestrare un modello. TensorFlow e PyTorch sono popolari framework di apprendimento automatico che forniscono interfacce intuitive e API flessibili per la creazione e l'addestramento di modelli di machine-learning. Questi framework supportano molte architetture di modelli e sono dotati di strumenti che semplificano il processo di addestramento.
10. **Riproducibilità:** La capacità di riprodurre i risultati del processo di training. La riproducibilità è importante per verificare la correttezza e la validità di un modello. Tuttavia, le variazioni dovute alle caratteristiche stocastiche della rete spesso rendono difficile riprodurre il comportamento preciso delle applicazioni in fase di addestramento, il che può rappresentare una sfida per il benchmarking.

Eseguendo il benchmarking per questi tipi di metriche, possiamo ottenere una visione completa delle prestazioni e dell'efficienza del processo di training da una prospettiva di sistema. Ciò può aiutare a identificare le aree di miglioramento e garantire che le risorse siano utilizzate in modo efficace.

11.4.4.3 I Task

Selezionare una manciata di task [attività] rappresentative per il benchmarking dei sistemi di machine learning è una sfida, perché l'apprendimento automatico viene applicato a vari domini con caratteristiche e requisiti unici. Ecco alcune delle sfide affrontate nella selezione di attività rappresentative:

1. **Diversità di Applicazioni:** L'apprendimento automatico viene utilizzato in numerosi campi, come sanità, finanza, elaborazione del linguaggio naturale, visione artificiale e molti altri. Ogni campo ha attività specifiche che potrebbero non essere rappresentative di altri campi. Ad esempio, le attività di classificazione delle immagini nella visione artificiale potrebbero non essere importanti per il rilevamento delle frodi finanziarie.
2. **Variabilità nei Tipi di Dati e nella Qualità:** Diverse attività richiedono tipi di dati diversi, come testo, immagini, video o dati numerici. La qualità e la disponibilità dei dati possono variare notevolmente tra le attività, rendendo difficile selezionare attività rappresentative delle sfide generali affrontate nell'apprendimento automatico.
3. **Complessità e Difficoltà delle Attività:** La complessità delle attività varia notevolmente. Alcune sono relativamente semplici, mentre altre sono molto complesse e richiedono modelli e tecniche sofisticate. Selezionare attività rappresentative che coprano le complessità riscontrate nell'apprendimento automatico è difficile.
4. **Problemi Etici e di Privacy:** Alcune attività possono riguardare dati sensibili o privati, come cartelle cliniche o informazioni personali. Queste attività possono presentare problemi etici e di privacy che devono essere affrontati, rendendole meno adatte come attività rappresentative per il benchmarking.
5. **Requisiti di Scalabilità e Risorse:** Attività diverse possono avere requisiti di scalabilità e risorse diverse. Alcune attività possono richiedere ampie risorse di calcolo, mentre altre possono essere eseguite con risorse minime. Selezionare attività che rappresentino i requisiti di risorse generali nell'apprendimento automatico è difficile.

6. **Metriche di Valutazione:** Le metriche utilizzate per valutare le prestazioni dei modelli di apprendimento automatico variano tra le attività. Alcune attività possono avere metriche di valutazione consolidate, mentre altre non hanno metriche chiare o standardizzate. Ciò può rendere difficile confrontare le prestazioni tra diverse attività.
7. **Generalizzabilità dei Risultati:** I risultati ottenuti dal benchmarking su un'attività specifica potrebbero non essere generalizzabili ad altre attività. Ciò significa che le prestazioni di un sistema di apprendimento automatico su un'attività selezionata potrebbero non essere indicative delle sue prestazioni su altre attività.

È importante considerare attentamente questi fattori quando si progettano benchmark per garantire che siano significativi e pertinenti per la vasta gamma di attività incontrate nel machine learning.

11.4.4.4 I Benchmark

Ecco alcuni lavori originali che hanno gettato le basi fondamentali per lo sviluppo di benchmark sistematici per l'addestramento di sistemi di apprendimento automatico.

MLPerf Training Benchmark

MLPerf è una suite di benchmark progettata per misurare le prestazioni di hardware, software e servizi di apprendimento automatico. Il benchmark di MLPerf Training (Mattson et al. 2020a) si concentra sul tempo necessario per addestrare i modelli a una metrica di qualità target. Include carichi di lavoro diversi, come classificazione delle immagini, rilevamento di oggetti, traduzione e apprendimento per rinforzo. Figura 11.2 evidenzia i miglioramenti delle prestazioni nelle versioni progressive dei benchmark di MLPerf Training, che hanno tutti superato la legge di Moore. L'utilizzo di trend di benchmarking standardizzati ci consente di mostrare rigorosamente la rapida evoluzione del ML computing.

Figura 11.2: Tendenze delle prestazioni di MLPerf Training. Fonte: Mattson et al. (2020a).

Metriche:

- Tempo di training per la qualità target
- Throughput (esempi al secondo)
- Utilizzo delle risorse (CPU, GPU, memoria, I/O del disco)

DAWNBench

DAWNBench (Coleman et al. 2019) è una suite di benchmark incentrata sul tempo di training end-to-end del deep learning e sulle prestazioni dell'inferenza. Include attività comuni come la classificazione delle immagini e la risposta alle domande.
Metriche:

- Tempo di training per la precisione target
- Latenza dell'inferenza
- Costo (in termini di risorse di cloud computing e storage)

Fathom

Fathom (Adolf et al. 2016) è un benchmark dell'Università di Harvard che valuta le prestazioni dei modelli di deep learning utilizzando un set diversificato di carichi di lavoro. Questi includono attività comuni come la classificazione delle immagini, il riconoscimento vocale e la modellazione del linguaggio.
Metriche:

- Operazioni al secondo (per misurare l'efficienza computazionale)
- Tempo di completamento per ogni carico di lavoro
- Larghezza di banda della memoria

11.4.4.5 Caso d'Uso di Esempio

Consideriamo uno scenario in cui vogliamo eseguire il benchmark dell'addestramento di un modello di classificazione delle immagini su una piattaforma hardware specifica.

1. **Task:** L'attività consiste nell'addestrare una rete neurale convoluzionale (CNN) per la classificazione delle immagini sul set di dati CIFAR-10.
2. **Benchmark:** Possiamo usare il benchmark di addestramento MLPerf per questa attività. Include un carico di lavoro di classificazione delle immagini pertinente alla nostra attività.
3. **Metriche:** Misureremo le seguenti metriche:

- Tempo di addestramento per raggiungere una precisione target del 90%.
- Throughput in termini di immagini elaborate al secondo.
- Utilizzo di GPU e CPU durante l'addestramento.

Misurando queste metriche, possiamo valutare le prestazioni e l'efficienza del processo di addestramento sulla piattaforma hardware selezionata. Queste informazioni possono quindi essere utilizzate per identificare potenziali colli di bottiglia o aree di miglioramento.

11.4.5 Benchmark di Inferenza

L'inferenza nell'apprendimento automatico si riferisce all'uso di un modello addestrato per fare previsioni su dati nuovi e mai visti prima. È la fase in cui il modello applica le conoscenze apprese per risolvere il problema per cui è stato progettato, come la classificazione di immagini, il riconoscimento vocale o la traduzione di testo.

11.4.5.1 Scopo

Quando creiamo modelli di machine learning, il nostro obiettivo finale è di distribuirli in applicazioni del mondo reale in cui possano fornire previsioni accurate e affidabili su dati nuovi e mai visti. Questo processo di utilizzo di un modello addestrato per fare previsioni è noto come inferenza. Le prestazioni reali di un modello di apprendimento automatico possono differire in modo significativo dalle sue prestazioni su set di dati di addestramento o validazione, il che rende l'inferenza di benchmarking un passaggio cruciale nello sviluppo e nell'implementazione di modelli di machine learning.
Il benchmarking dell'inferenza ci consente di valutare quanto bene un modello di apprendimento automatico funziona in scenari del mondo reale. Questa valutazione

garantisce che il modello sia pratico e affidabile quando distribuito in applicazioni, fornendo una comprensione più completa del comportamento del modello con dati reali. Inoltre, il benchmarking può aiutare a identificare potenziali colli di bottiglia o limitazioni nelle prestazioni del modello. Ad esempio, se un modello impiega troppo tempo per dedurre, potrebbe non essere pratico per applicazioni in tempo reale come la guida autonoma o gli assistenti vocali.
L'efficienza delle risorse è un altro aspetto critico dell'inferenza, poiché può essere computazionalmente intensiva e richiedere memoria e potenza di elaborazione significative. Il benchmarking aiuta a garantire che il modello sia efficiente per quanto riguarda l'utilizzo delle risorse, il che è particolarmente importante per i dispositivi edge con capacità computazionali limitate, come smartphone o dispositivi IoT. Inoltre, il benchmarking ci consente di confrontare le prestazioni del nostro modello con quelli concorrenti o versioni precedenti dello stesso modello. Questo confronto è essenziale per prendere decisioni informate su quale modello implementare in un'applicazione specifica.
Infine, è fondamentale garantire che le previsioni del modello non siano solo accurate, ma anche coerenti tra diversi dati. Il benchmarking aiuta a verificare l'accuratezza e la coerenza del modello, assicurando che soddisfi i requisiti dell'applicazione. Valuta inoltre la robustezza del modello, assicurando che possa gestire la variabilità dei dati del mondo reale e comunque fare previsioni accurate.

11.4.5.2 Metriche

1. **Precisione:** La precisione è una delle metriche più importanti quando si confrontano i modelli di machine learning. Quantifica la percentuale di previsioni corrette effettuate dal modello rispetto ai valori o alle etichette reali. Ad esempio, se un modello di rilevamento dello spam riesce a classificare correttamente 95 messaggi e-mail su 100, la sua precisione verrebbe calcolata al 95%.
2. **Latenza:** La latenza è una metrica delle prestazioni che calcola il ritardo o l'intervallo di tempo tra la ricezione dell'input e la produzione dell'output corrispondente da parte del sistema di apprendimento automatico. Un esempio che descrive chiaramente la latenza è un'applicazione di traduzione in tempo reale; se esiste un ritardo di mezzo secondo dal momento in cui un utente inserisce una frase al momento in cui l'app visualizza il testo tradotto, la latenza del sistema è di 0.5 secondi.
3. **Latency-Bounded Throughput:** Il throughput limitato dalla latenza è una metrica preziosa che combina gli aspetti di latenza e throughput, misurando il throughput massimo di un sistema pur rispettando un vincolo di latenza specificato. Ad esempio, in un'applicazione di streaming video che utilizza un modello di apprendimento automatico per generare e visualizzare automaticamente i sottotitoli, il throughput limitato dalla latenza misurerebbe quanti frame video il sistema può elaborare al secondo (throughput) garantendo al contempo che i sottotitoli vengano visualizzati con un ritardo non superiore a 1 secondo (latenza). Questa metrica è particolarmente importante nelle applicazioni in tempo reale in cui soddisfare i requisiti di latenza è fondamentale per l'esperienza utente.
4. **Throughput:** Il throughput valuta la capacità del sistema misurando il numero di inferenze o previsioni che un modello di apprendimento automatico può gestire entro un'unità di tempo specifica. Si consideri un sistema di riconoscimento vocale che utilizza una Recurrent Neural Network (RNN) come modello sottostante; se questo sistema riesce a elaborare e comprendere 50 diverse clip audio in un minuto, allora la sua velocità di elaborazione è di 50 clip al minuto.
5. **Efficienza energetica:** L'efficienza energetica è una metrica che determina la quantità di energia consumata dal modello di apprendimento automatico per eseguire una

singola inferenza. Un esempio lampante di ciò sarebbe un modello di elaborazione del linguaggio naturale basato su un'architettura di rete Transformer; se utilizza 0,1 Joule di energia per tradurre una frase dall'inglese al francese, la sua efficienza energetica è misurata a 0,1 Joule per inferenza.

6. **Utilizzo della memoria:** L'utilizzo della memoria quantifica il volume di RAM necessario a un modello di apprendimento automatico per svolgere attività di inferenza. Un esempio rilevante per illustrare questo sarebbe un sistema di riconoscimento facciale basato su una CNN; se un tale sistema richiede 150 MB di RAM per elaborare e riconoscere i volti all'interno di un'immagine, il suo utilizzo della memoria è di 150 MB.

11.4.5.3 I Task

Le sfide nella scelta di attività rappresentative per il benchmarking dei sistemi di apprendimento automatico inferenziale sono, in generale, piuttosto simili alla tassonomia che abbiamo fornito per la il training. Tuttavia, per essere pignoli, discutiamone nel contesto dei sistemi di machine learning inferenziale.

1. **Diversità di Applicazioni:** L'apprendimento automatico inferenziale è impiegato in numerosi domini come sanità, finanza, intrattenimento, sicurezza e altro. Ogni dominio ha attività uniche e ciò che è rappresentativo in un dominio potrebbe non esserlo in un altro. Ad esempio, un'attività di inferenza per prevedere i prezzi delle azioni nel dominio finanziario potrebbe differire dalle attività di riconoscimento delle immagini nel dominio medico.
2. **Variabilità nei Tipi di Dati:** Diverse attività di inferenza richiedono diversi tipi di dati: testo, immagini, video, dati numerici, ecc. Assicurarsi che i benchmark affrontino l'ampia varietà di tipi di dati utilizzati nelle applicazioni del mondo reale è una sfida. Ad esempio, i sistemi di riconoscimento vocale elaborano dati audio, che sono molto diversi dai dati visivi elaborati dai sistemi di riconoscimento facciale.
3. **Complessità delle Attività:** La complessità delle attività di inferenza può variare enormemente, da attività di classificazione di base ad attività complesse che richiedono modelli all'avanguardia. Ad esempio, distinguere tra due categorie (classificazione binaria) è in genere più semplice che rilevare centinaia di tipi di oggetti in una scena affollata.
4. **Requisiti in Tempo Reale:** Alcune applicazioni richiedono risposte immediate o in tempo reale, mentre altre possono consentire un certo ritardo. Nella guida autonoma, il rilevamento degli oggetti in tempo reale e il processo decisionale sono fondamentali, mentre un motore di raccomandazione per un sito Web di shopping potrebbe tollerare lievi ritardi.
5. **Problemi di Scalabilità:** Data la variabilità della scala delle applicazioni, dai dispositivi edge ai server basati su cloud, le attività devono rappresentare i diversi ambienti di elaborazione in cui si verifica l'inferenza. Ad esempio, un'attività di inferenza in esecuzione sulle risorse limitate di uno smartphone è diversa da un potente server cloud.
6. **Diversità delle Metriche di Valutazione:** Le metriche utilizzate per valutare le prestazioni possono variare in modo significativo a seconda dell'attività. Trovare un terreno comune o una metrica universalmente accettata per attività diverse è una sfida. Ad esempio, la precisione e il "recall" [richiamo] potrebbero essere vitali per un'attività di diagnosi medica, mentre la produttività (inferenze al secondo) potrebbero essere più cruciali per le attività di elaborazione video.
7. **Problemi Etici e di Privacy:** Esistono problemi relativi all'etica e alla privacy, soprattutto in aree sensibili come il riconoscimento facciale o l'elaborazione dei dati

personali. Questi problemi possono influire sulla selezione e sulla natura delle attività utilizzate per il benchmarking. Ad esempio, l'utilizzo di dati facciali reali per il benchmarking può sollevare problemi di privacy, mentre i dati sintetici potrebbero non replicare le sfide del mondo reale.

8. **Diversità Hardware:** Con un'ampia gamma di dispositivi, da GPU, CPU e TPU ad ASIC personalizzati utilizzati per l'inferenza, garantire che le attività siano rappresentative su hardware diversi è una sfida. Ad esempio, un'attività ottimizzata per l'inferenza su una GPU potrebbe avere prestazioni non ottimali su un dispositivo edge.

11.4.5.4 I Benchmark

Ecco alcuni lavori originali che hanno gettato le basi fondamentali per lo sviluppo di benchmark sistematici per sistemi di apprendimento automatico inferenziale.

MLPerf Inference Benchmark: MLPerf Inference è una suite di benchmark completa che valuta le prestazioni dei modelli di apprendimento automatico durante la fase di inferenza. Comprende una varietà di carichi di lavoro, tra cui classificazione delle immagini, rilevamento di oggetti ed elaborazione del linguaggio naturale, con l'obiettivo di fornire metriche standardizzate e approfondite per la valutazione di diversi sistemi di inferenza. Le sue metriche includono:
MLPerf Inference è una suite di benchmark completa che valuta le prestazioni dei modelli di apprendimento automatico durante la fase di inferenza. Comprende una varietà di carichi di lavoro, tra cui classificazione delle immagini, rilevamento di oggetti ed elaborazione del linguaggio naturale, con l'obiettivo di fornire metriche standardizzate e approfondite per la valutazione di diversi sistemi di inferenza.
Metriche:

- Tempo di inferenza
- Latenza
- Throughput [Produttività]
- Precisione
- Consumo energetico

AI Benchmark: AI Benchmark è uno strumento di benchmarking che valuta le prestazioni dei modelli di intelligenza artificiale e apprendimento automatico su dispositivi mobili e piattaforme di edge computing. Include test per attività di classificazione delle immagini, rilevamento di oggetti ed elaborazione del linguaggio naturale, fornendo un'analisi dettagliata delle prestazioni di inferenza su diverse piattaforme hardware. Le sue metriche includono:
AI Benchmark è uno strumento di benchmarking che valuta le prestazioni dei modelli di intelligenza artificiale e di apprendimento automatico su dispositivi mobili e piattaforme di edge computing. Include test per attività di classificazione delle immagini, rilevamento di oggetti ed elaborazione del linguaggio naturale, fornendo un'analisi dettagliata delle prestazioni di inferenza su diverse piattaforme hardware.
Metriche:

- Tempo di inferenza
- Latenza
- Consumo energetico
- Utilizzo della memoria
- Throughput [Produttività]

Toolkit OpenVINO: Il toolkit OpenVINO fornisce uno strumento di benchmark per misurare le prestazioni dei modelli di apprendimento profondo per varie attività, come la classificazione delle immagini, il rilevamento degli oggetti e il riconoscimento facciale, su hardware Intel.

Offre approfondimenti dettagliati sulle prestazioni di inferenza dei modelli su diverse configurazioni hardware. Le sue metriche includono:
Metriche:

- Tempo di inferenza
- Throughput [Produttività]
- Latenza
- Utilizzo di CPU e GPU

11.4.5.5 Caso d'Uso di Esempio

Consideriamo uno scenario in cui vogliamo valutare le prestazioni di inferenza di un modello di rilevamento di oggetti su uno specifico dispositivo edge.
Task: L'attività consiste nell'eseguire il rilevamento di oggetti in tempo reale su flussi video, rilevando e identificando oggetti quali veicoli, pedoni e segnali stradali.
Benchmark: Possiamo utilizzare AI Benchmark per questa attività in quanto valuta le prestazioni di inferenza sui dispositivi edge, il che si adatta al nostro scenario.
Metriche: Misureremo le seguenti metriche:

- Tempo di inferenza per elaborare ogni fotogramma video
- Latenza per generare i "bounding box" per gli oggetti rilevati
- Consumo energetico durante il processo di inferenza
- Produttività in termini di fotogrammi video elaborati al secondo

Misurando queste metriche, possiamo valutare le prestazioni del modello di rilevamento di oggetti sul dispositivo edge e identificare eventuali colli di bottiglia o aree di ottimizzazione per migliorare le capacità di elaborazione in tempo reale.

Esercizio 11.2: Benchmark di Inferenza - MLPerf

Prepararsi a mettere alla prova i propri modelli di intelligenza artificiale! MLPerf è come le Olimpiadi per le prestazioni del machine learning. In questo Colab, utilizzeremo un toolkit chiamato CK per eseguire benchmark MLPerf ufficiali, misurare la velocità e l'accuratezza di un proprio modello e persino utilizzare TVM per dargli una spinta super veloce. Pronti a vedere il modello vincere la sua medaglia?

Open in Colab

11.4.6 Misura dell'Efficienza Energetica

Con l'espansione delle capacità di apprendimento automatico, sia nel training che nell'inferenza, le preoccupazioni relative all'aumento del consumo energetico e al suo impatto ecologico si sono intensificate. Affrontare la sostenibilità dei sistemi ML, un argomento esplorato più approfonditamente nel capitolo IA Sostenibile, è quindi diventata una priorità fondamentale. Questa attenzione alla sostenibilità ha portato allo sviluppo di benchmark standardizzati progettati per misurare con precisione l'efficienza energetica. Tuttavia, la standardizzazione di queste metodologie pone delle sfide dovute alla necessità di adattarsi a scale molto diverse, dal consumo di microwatt dei dispositivi TinyML alle richieste di megawatt dei sistemi di training dei data center. Inoltre, per garantire che il benchmarking sia equo e riproducibile è necessario adattarsi alla vasta gamma di configurazioni hardware e architetture in uso oggi.
Un esempio è la metodologia di benchmarking MLPerf Power (Tschand et al. 2024), che affronta queste sfide adattando le metodologie per data center, edge inference e tiny inference systems, misurando al contempo il consumo energetico nel modo più completo possibile per ogni scala. Questa metodologia si adatta a una varietà di hardware, dalle CPU

generiche agli acceleratori AI specializzati, mantenendo principi di misurazione uniformi per garantire che i confronti siano equi e accurati su diverse piattaforme.
Figura 11.3 illustra i limiti di misurazione dell'alimentazione per diverse scale di sistema, dai dispositivi TinyML ai nodi di inferenza e ai rack di training. Ciascun esempio evidenzia i componenti all'interno del limite di misurazione e quelli al di fuori di esso. Questa configurazione consente una riflessione accurata dei veri costi energetici associati all'esecuzione di carichi di lavoro ML in vari scenari del mondo reale e garantisce che il benchmark catturi l'intero spettro di consumo energetico.

Figura 11.3: Diagramma di misurazione del sistema MLPerf Power. Fonte: Tschand et al. (2024).

È importante notare che l'ottimizzazione di un sistema per le prestazioni potrebbe non portare all'esecuzione più efficiente dal punto di vista energetico. Spesso, sacrificare una piccola quantità di prestazioni o accuratezza può portare a guadagni significativi nell'efficienza energetica, evidenziando l'importanza di un benchmarking accurato delle metriche della potenza. Le future intuizioni dal benchmarking dell'efficienza energetica e della sostenibilità ci consentiranno di ottimizzare per sistemi ML più sostenibili.

11.4.7 Esempio di Benchmark

Per illustrare correttamente i componenti di un benchmark di sistema, possiamo esaminare il benchmark di individuazione delle parole chiave in MLPerf Tiny e spiegare la motivazione alla base di ogni decisione.

11.4.7.1 Task

L'individuazione delle parole chiave è stata selezionata come attività perché è un caso d'uso comune in TinyML che è stato ben consolidato per anni. Inoltre, l'hardware tipico utilizzato per l'individuazione delle parole chiave differisce sostanzialmente dalle offerte di altri benchmark, come l'attività di riconoscimento vocale di MLPerf Inference.

11.4.7.2 Il Dataset

Google Speech Commands (Warden 2018) è stato selezionato come il miglior dataset per rappresentare l'attività. Il dataset è ben consolidato nella comunità di ricerca e ha una licenza permissiva, che consente di utilizzarlo facilmente in un benchmark.

11.4.7.3 Modello

Il componente principale successivo è il modello, che fungerà da carico di lavoro primario per il benchmark. Il modello dovrebbe essere ben consolidato come soluzione per l'attività selezionata piuttosto che una soluzione all'avanguardia. Il modello selezionato è un semplice modello di convoluzione separabile in profondità. Questa architettura non è la soluzione all'avanguardia per l'attività, ma è ben consolidata e non progettata per una piattaforma hardware specifica come molte soluzioni all'avanguardia. Nonostante sia un benchmark di inferenza, stabilisce anche una ricetta di training di riferimento per essere completamente riproducibile e trasparente.

11.4.7.4 Metriche

La latenza è stata selezionata come metrica primaria per il benchmark, poiché i sistemi di individuazione delle parole chiave devono reagire rapidamente per mantenere la soddisfazione dell'utente. Inoltre, dato che i sistemi TinyML sono spesso alimentati a batteria, il consumo energetico viene misurato per garantire l'efficienza della piattaforma hardware. L'accuratezza del modello viene misurata anche per garantire che le ottimizzazioni applicate da un submitter, come la quantizzazione, non degradino l'accuratezza oltre una soglia.

11.4.7.5 Benchmark Harness

MLPerf Tiny utilizza EEMBCs EnergyRunner benchmark harness per caricare gli input nel modello e isolare e misurare il consumo energetico del dispositivo. Quando si misura il consumo energetico, è fondamentale selezionare un "harness" [imbracatura] che sia accurato ai livelli di potenza previsti dei dispositivi sottoposti a test e sufficientemente semplice da non diventare un peso per i partecipanti al benchmark.

11.4.7.6 La Baseline

Gli invii di baseline sono fondamentali per contestualizzare i risultati e come punto di riferimento per aiutare i partecipanti a iniziare. L'invio di base dovrebbe dare priorità alla semplicità e alla leggibilità rispetto alle prestazioni avanzate. L'individuazione della parola chiave della baseline utilizza un microcontrollore STM standard come hardware e TensorFlow Lite come microcontrollore (David et al. 2021) come framework di inferenza.

11.4.8 Sfide e Limitazioni

Sebbene il benchmarking fornisca una metodologia strutturata per la valutazione delle prestazioni in domini complessi come l'intelligenza artificiale e l'informatica, il processo pone anche diverse sfide. Se non affrontati correttamente, questi ostacoli possono minare la credibilità e l'accuratezza dei risultati del benchmarking. Alcune delle difficoltà predominanti affrontate nel benchmarking includono quanto segue:

- **Copertura incompleta del problema:** Le attività di benchmarking potrebbero non rappresentare completamente lo spazio del problema. Ad esempio, i set di dati di classificazione delle immagini comuni come CIFAR-10 hanno una diversità limitata nei tipi di immagini. Gli algoritmi ottimizzati per tali benchmark potrebbero non riuscire a generalizzare bene con i set di dati del mondo reale.
- **Insignificanza statistica:** I benchmark devono avere prove e campioni di dati sufficienti per produrre risultati statisticamente significativi. Ad esempio, il benchmarking di un modello OCR su solo poche scansioni di testo potrebbe non catturare adeguatamente i suoi veri tassi di errore.
- **Riproducibilità limitata:** Variazioni di hardware, versioni software, basi di codice e altri fattori possono ridurre la riproducibilità dei risultati di benchmark. MLPerf affronta questo problema fornendo implementazioni di riferimento e specifiche ambientali.
- **Disallineamento con gli obiettivi finali:** I benchmark che si concentrano solo su metriche di velocità o accuratezza possono disallineare gli obiettivi reali come costi ed efficienza energetica. I benchmark devono riflettere tutti gli assi prestazionali critici.
- **Rapida obsolescenza:** A causa del rapido ritmo dei progressi nell'intelligenza artificiale e nell'informatica, i benchmark e i loro set di dati possono rapidamente diventare obsoleti. Mantenere benchmark aggiornati è quindi una sfida persistente.

Ma di tutte queste, la sfida più importante è l'ingegneria dei benchmark.

11.4.8.1 Lotteria Hardware

La lotteria hardware, descritta per la prima volta da Hooker (2021), si riferisce alla situazione in cui il successo o l'efficienza di un modello di apprendimento automatico sono

significativamente influenzati dalla sua compatibilità con l'hardware sottostante (Chu et al. 2021). Alcuni modelli hanno prestazioni eccezionali non perché sono intrinsecamente superiori, ma perché sono ottimizzati per caratteristiche hardware specifiche, come le capacità di elaborazione parallela delle unità di elaborazione grafica (GPU) o delle unità di elaborazione tensoriale (TPU).
Ad esempio, Figura 11.4 confronta le prestazioni dei modelli su diverse piattaforme hardware. I modelli multi-hardware mostrano risultati comparabili a "MobileNetV3 Large min" sia sulle configurazioni CPU uint8 che GPU. Tuttavia, questi modelli multi-hardware dimostrano miglioramenti significativi delle prestazioni rispetto alla baseline MobileNetV3 Large quando eseguiti su hardware EdgeTPU e DSP. Ciò sottolinea l'efficienza variabile dei modelli multi-hardware in ambienti di elaborazione specializzati.

Figura 11.4: Compromessi tra precisione e latenza di più modelli ML e modalità di funzionamento su vari hardware. Fonte: Chu et al. (2021)

La lotteria hardware può introdurre sfide e pregiudizi nel benchmarking dei sistemi di apprendimento automatico, poiché le prestazioni del modello non dipendono esclusivamente dall'architettura o dall'algoritmo del modello ma anche dalla compatibilità e dalle sinergie con l'hardware sottostante. Ciò può rendere difficile confrontare equamente diversi modelli e identificare il modello migliore in base ai suoi meriti intrinseci. Può anche portare a una situazione in cui la comunità converge su modelli che sono adatti all'hardware più diffuso del momento, trascurando potenzialmente altri modelli che potrebbero essere superiori ma incompatibili con le attuali tendenze hardware.

11.4.8.2 Benchmark Engineering

La lotteria hardware si verifica quando un modello di apprendimento automatico funziona in modo eccezionalmente bene o male su una configurazione hardware specifica a causa di compatibilità o incompatibilità impreviste. Il modello non è esplicitamente progettato o ottimizzato per quell'hardware specifico dagli sviluppatori o dagli ingegneri; piuttosto, capita che si allinei o (non si allinei) con le capacità o le limitazioni dell'hardware. In questo caso, le prestazioni del modello sull'hardware sono un prodotto della coincidenza piuttosto che della progettazione.
Contrariamente alla lotteria hardware accidentale, il benchmark engineering implica l'ottimizzazione o la progettazione deliberata di un modello di apprendimento automatico per funzionare eccezionalmente bene su hardware specifico, spesso per vincere benchmark o competizioni. Questa ottimizzazione intenzionale potrebbe includere la modifica

dell'architettura, degli algoritmi o dei parametri del modello per sfruttare appieno le funzionalità e le capacità dell'hardware.

11.4.8.3 Problema

Il benchmark engineering si riferisce alla modifica o all'ottimizzazione di un sistema di intelligenza artificiale per ottimizzare le prestazioni su test di benchmark specifici, spesso a scapito della generalizzabilità o delle prestazioni nel mondo reale. Ciò può includere la regolazione di iperparametri, dati di training o altri aspetti del sistema specificamente per ottenere punteggi elevati sulle metriche di benchmark senza necessariamente migliorare la funzionalità o l'utilità complessiva del sistema.

La motivazione alla base dell'ingegneria dei benchmark spesso deriva dal desiderio di ottenere punteggi di prestazioni elevate per scopi di marketing o competitivi. Punteggi di benchmark elevati possono dimostrare la superiorità di un sistema di intelligenza artificiale rispetto ai concorrenti e possono essere un argomento chiave per la vendita per potenziali utenti o investitori. Questa pressione per ottenere buoni risultati nei benchmark a volte porta a dare priorità alle ottimizzazioni specifiche del benchmark rispetto a miglioramenti più olistici del sistema.

Può comportare diversi rischi e sfide. Uno dei rischi principali è che il sistema di intelligenza artificiale possa funzionare meglio nelle applicazioni del mondo reale rispetto a quanto suggeriscono i punteggi di benchmark. Ciò può portare a insoddisfazione dell'utente, danni alla reputazione e potenziali problemi di sicurezza o etici. Inoltre, l'ingegneria dei benchmark può contribuire a una mancanza di trasparenza e responsabilità nella comunità dell'intelligenza artificiale, poiché può essere difficile discernere quanta parte delle prestazioni di un sistema di intelligenza artificiale sia dovuta a miglioramenti genuini rispetto a ottimizzazioni specifiche del benchmark.

La comunità AI deve dare priorità alla trasparenza e alla responsabilità per mitigare i rischi associati all'ingegneria dei benchmark. Ciò può includere la divulgazione di eventuali ottimizzazioni o modifiche apportate specificamente per i test di benchmark e la fornitura di valutazioni più complete dei sistemi AI che includono metriche delle prestazioni del mondo reale e punteggi di benchmark. I ricercatori e gli sviluppatori devono dare priorità a miglioramenti olistici dei sistemi AI che ne migliorino la generalizzabilità e la funzionalità in varie applicazioni anziché concentrarsi esclusivamente su ottimizzazioni specifiche del benchmark.

11.4.8.4 Problemi

Uno dei problemi principali dell'ingegneria del benchmark è che può compromettere le prestazioni reali dei sistemi di intelligenza artificiale. Quando gli sviluppatori si concentrano sull'ottimizzazione dei loro sistemi per ottenere punteggi elevati in specifici test di benchmark, potrebbero trascurare altri importanti aspetti delle prestazioni del sistema, cruciali nelle applicazioni del mondo reale. Ad esempio, un sistema di intelligenza artificiale progettato per il riconoscimento delle immagini potrebbe essere progettato per funzionare eccezionalmente bene in un test di benchmark che include un set specifico di immagini, ma necessita di aiuto per riconoscere accuratamente immagini leggermente diverse da quelle nel set di test.

Un'altra area di miglioramento con l'ingegneria di benchmark è che può comportare sistemi di intelligenza artificiale privi di generalizzabilità. In altre parole, mentre il sistema può funzionare bene nel test di benchmark, potrebbe aver bisogno di aiuto per gestire una vasta gamma di input o scenari. Ad esempio, un modello di intelligenza artificiale sviluppato per l'elaborazione del linguaggio naturale potrebbe essere progettato per ottenere punteggi elevati in un test di benchmark che include un tipo specifico di testo, ma non riesce a elaborare accuratamente il testo che non rientra in quel tipo specifico.

Può anche portare a risultati fuorvianti. Quando i sistemi di intelligenza artificiale sono progettati per funzionare bene nei test di benchmark, i risultati potrebbero non riflettere accuratamente le reali capacità del sistema. Questo può essere problematico per gli utenti o gli investitori che si affidano ai punteggi di benchmark per prendere decisioni informate su quali sistemi di intelligenza artificiale utilizzare o in cui investire. Ad esempio, un sistema di intelligenza artificiale progettato per ottenere punteggi elevati in un test di benchmark per il riconoscimento vocale potrebbe dover essere più in grado di riconoscere accuratamente il parlato in situazioni reali, portando gli utenti o gli investitori a prendere decisioni basate su informazioni imprecise.

11.4.8.5 Attenuazione

Esistono diversi modi per mitigare l'ingegneria dei benchmark. La trasparenza nel processo di benchmarking è fondamentale per mantenere l'accuratezza e l'affidabilità dei benchmark. Ciò implica la divulgazione chiara delle metodologie, dei set di dati e dei criteri di valutazione utilizzati nei test di benchmark, nonché di eventuali ottimizzazioni o modifiche apportate al sistema di intelligenza artificiale ai fini del benchmark.

Un modo per ottenere trasparenza è attraverso l'uso di benchmark open source. I benchmark open source vengono resi disponibili al pubblico, consentendo a ricercatori, sviluppatori e altre parti interessate di esaminarli, criticarli e contribuire, garantendone così l'accuratezza e l'affidabilità. Questo approccio collaborativo facilita anche la condivisione delle "best practice" e lo sviluppo di benchmark più solidi e completi.

Un esempio è MLPerf Tiny. È un framework open source progettato per semplificare il confronto di diverse soluzioni nel mondo di TinyML. Il suo design modulare consente di sostituire i componenti per il confronto o il miglioramento. Le implementazioni di riferimento, mostrate in verde e arancione in Figura 11.5, fungono da base per i risultati. TinyML spesso necessita di ottimizzazione nell'intero sistema e gli utenti possono contribuire concentrandosi su parti specifiche, come la quantizzazione. Il design modulare del benchmark consente agli utenti di mostrare i propri contributi e il vantaggio competitivo modificando un'implementazione di riferimento. In breve, MLPerf Tiny offre un modo flessibile e modulare per valutare e migliorare le applicazioni TinyML, semplificando il confronto e il miglioramento di diversi aspetti della tecnologia.

Figura 11.5: Design modulare di MLPerf Tiny. Fonte: Mattson et al. (2020a).

Un altro metodo per ottenere trasparenza è attraverso la revisione paritaria dei benchmark. Ciò comporta che esperti indipendenti esaminino e convalidino la metodologia, i set di dati e i risultati del benchmark per garantirne la credibilità e l'affidabilità. La revisione paritaria può fornire un mezzo prezioso per verificare l'accuratezza dei test di benchmark e contribuire a creare fiducia nei risultati.

La standardizzazione dei benchmark è un'altra importante soluzione per mitigare l'ingegneria dei benchmark. I benchmark standardizzati forniscono un quadro comune per la valutazione dei sistemi di intelligenza artificiale, garantendo coerenza e comparabilità tra diversi sistemi e applicazioni. Ciò può essere ottenuto sviluppando standard e "best practice" per l'intero settore per il benchmarking e tramite metriche e criteri di valutazione comuni.

Anche la verifica da parte di terze parti dei risultati può essere preziosa per mitigare l'ingegneria dei benchmark. Ciò comporta che una terza parte indipendente verifichi i risultati di un test di benchmark per garantirne la credibilità e l'affidabilità. La verifica di terze parti può creare fiducia nei risultati e fornire un mezzo prezioso per convalidare le prestazioni e le capacità dei sistemi di intelligenza artificiale.

11.5 Benchmarking del Modello

Il benchmarking dei modelli di machine learning è importante per determinare l'efficacia e l'efficienza di vari algoritmi di apprendimento automatico nella risoluzione di compiti o problemi specifici. Analizzando i risultati ottenuti dal benchmarking, sviluppatori e ricercatori possono identificare i punti di forza e di debolezza dei loro modelli, portando a decisioni più informate sulla selezione del modello e su un'ulteriore ottimizzazione.

L'evoluzione e il progresso dei modelli di apprendimento automatico sono intrinsecamente collegati alla disponibilità e alla qualità dei set di dati. Nell'apprendimento automatico, i dati fungono da materia prima che alimenta gli algoritmi, consentendo loro di apprendere, adattarsi e, in definitiva, eseguire compiti che erano tradizionalmente di dominio degli esseri umani. Pertanto, è importante comprendere questa storia.

11.5.1 Contesto Storico

I dataset di apprendimento automatico hanno una storia ricca e si sono evoluti in modo significativo nel corso degli anni, crescendo in dimensioni, complessità e diversità per soddisfare le richieste sempre crescenti del settore. Diamo un'occhiata più da vicino a questa evoluzione, partendo da uno dei primi e più iconici set di dati: MNIST.

11.5.1.1 MNIST (1998)

Il dataset MNIST, creato da Yann LeCun, Corinna Cortes e Christopher J.C. Burges nel 1998, può essere considerato una pietra miliare nella storia dei dataset di machine learning. Comprende 70.000 immagini in scala di grigi da 28x28 pixel etichettate di cifre scritte a mano (0-9). MNIST è stato ampiamente utilizzato per il benchmarking degli algoritmi nell'elaborazione delle immagini e nell'apprendimento automatico come punto di partenza per molti ricercatori e professionisti. Figura 11.6 mostra alcuni esempi di cifre scritte a mano.

Figura 11.6: Cifre scritte a mano in MNIST. Fonte: Suvanjanprasai

11.5.1.2 ImageNet (2009)

Facciamo un salto al 2009 e vediamo l'introduzione di ImageNet, che ha segnato un balzo significativo nella scala e nella complessità dei dataset. ImageNet è composto da oltre 14 milioni di immagini etichettate che abbracciano più di 20.000 categorie. Fei-Fei Li e il suo team lo hanno sviluppato per far progredire il riconoscimento degli oggetti e la ricerca sulla visione artificiale. Il dataset è diventato sinonimo della ImageNet Large Scale Visual Recognition Challenge (LSVRC), una competizione annuale cruciale nello sviluppo di modelli di deep learning, tra cui il famoso AlexNet nel 2012.

11.5.1.3 COCO (2014)

Il Common Objects in Context (COCO) dataset (T.-Y. Lin et al. 2014), rilasciato nel 2014, ha ulteriormente ampliato il panorama dei set di dati di apprendimento automatico introducendo un set più ricco di annotazioni. COCO è costituito da immagini contenenti scene complesse con più oggetti e ogni immagine è annotata con riquadri di delimitazione degli oggetti, maschere di segmentazione e didascalie, come mostrato in Figura 11.7. Questo set di dati è stato determinante nel far progredire la ricerca nel rilevamento degli oggetti, nella segmentazione e nella didascalia delle immagini.

Figura 11.7: Immagini di esempio dal set di dati COCO. Fonte: Coco

11.5.1.4 GPT-3 (2020)

Sebbene gli esempi sopra riportati si concentrino principalmente sui dataset di immagini, si sono verificati anche sviluppi significativi nei dataset di testo. Un esempio degno di nota è GPT-3 (Brown et al. 2020), sviluppato da OpenAI. GPT-3 è un modello linguistico addestrato su testo Internet eterogeneo. Sebbene il dataset utilizzato per addestrare GPT-3 non sia disponibile al pubblico, il modello stesso, costituito da 175 miliardi di parametri, è una testimonianza della scala e della complessità dei moderni dataset e modelli di apprendimento automatico.

11.5.1.5 Presente e Futuro

Oggi disponiamo di una pletora di dataset che abbracciano vari domini, tra cui sanità, finanza, scienze sociali e altro ancora. Le seguenti caratteristiche ci aiutano a classificare lo spazio e la crescita dei dataset di apprendimento automatico che alimentano lo sviluppo del modello.

1. **Diversità dei Set di Dati:** La varietà di set di dati disponibili per ricercatori e ingegneri si è ampliata notevolmente, coprendo molti campi, tra cui l'elaborazione del linguaggio naturale, il riconoscimento delle immagini e altro ancora. Questa diversità ha alimentato lo sviluppo di modelli di apprendimento automatico specializzati, su misura per attività specifiche, come la traduzione, il riconoscimento vocale e il riconoscimento facciale.
2. **Volume di Dati:** L'enorme volume di dati che è diventato disponibile nell'era digitale ha anche svolto un ruolo cruciale nel progresso dei modelli di apprendimento automatico. I grandi set di dati consentono ai modelli di catturare la complessità e le sfumature dei fenomeni del mondo reale, portando a previsioni più accurate e affidabili.
3. **Qualità e Pulizia dei Dati:** La qualità dei dati è un altro fattore critico che influenza le prestazioni dei modelli di apprendimento automatico. Set di dati puliti, ben etichettati e imparziali sono essenziali per modelli di addestramento solidi ed equi.
4. **Accesso Aperto ai Dati:** La disponibilità di set di dati "open-access" ha contribuito in modo significativo anche al progresso dell'apprendimento automatico. I dati "aperti" consentono ai ricercatori di tutto il mondo di collaborare, condividere approfondimenti e basarsi sul lavoro degli altri, portando a un'innovazione più rapida e allo sviluppo di modelli più avanzati.
5. **Problemi di Etica e Privacy:** Man mano che i set di dati crescono in dimensioni e complessità, le considerazioni etiche e i problemi di privacy diventano sempre più importanti. È in corso un dibattito sull'equilibrio tra lo sfruttamento dei dati per i progressi dell'apprendimento automatico e la protezione dei diritti alla privacy degli individui.

Lo sviluppo di modelli di apprendimento automatico si basa in larga misura sulla disponibilità di set di dati diversificati, grandi, di alta qualità e ad accesso libero. Mentre andiamo avanti, affrontare le considerazioni etiche e le preoccupazioni sulla privacy associate all'uso di grandi set di dati è fondamentale per garantire che le tecnologie di apprendimento automatico siano vantaggiose per la società. C'è una crescente consapevolezza che i dati agiscono come carburante per l'apprendimento automatico, guidando e alimentando lo sviluppo di modelli di apprendimento automatico. Di conseguenza, si sta ponendo maggiore attenzione sullo sviluppo dei set di dati stessi. Esploreremo questo aspetto in modo più dettagliato nella sezione del benchmarking dei dati.

11.5.2 Metriche del Modello

La valutazione del modello di machine learning si è evoluta da un focus ristretto sulla precisione a un approccio più completo che considera una serie di fattori, da considerazioni etiche e applicabilità nel mondo reale a vincoli pratici come dimensioni ed efficienza del modello. Questo cambiamento riflette la maturazione del campo poiché i modelli di apprendimento automatico vengono sempre più applicati in scenari reali diversi e complessi.

11.5.2.1 Precisione

La precisione è una delle metriche più intuitive e comunemente utilizzate per valutare i modelli di apprendimento automatico. In sostanza, la precisione misura la percentuale di previsioni corrette effettuate dal modello rispetto a tutte le previsioni. Ad esempio, immaginiamo di aver sviluppato un modello di apprendimento automatico per classificare le immagini come contenenti o meno un gatto. Se testiamo questo modello su un set di dati di 100 immagini e ne identifica correttamente 90, calcoleremmo la sua precisione al 90%.
Nelle fasi iniziali dell'apprendimento automatico, la precisione era spesso la metrica principale, se non l'unica, considerata quando si valutavano le prestazioni del modello. Ciò è comprensibile, data la sua natura semplice e la facilità di interpretazione. Tuttavia, con il progredire del settore, i limiti del fare affidamento esclusivamente sulla precisione sono diventati più evidenti.
Si consideri l'esempio di un modello di diagnosi medica con una precisione del 95%. Sebbene a prima vista possa sembrare impressionante, dobbiamo guardare più a fondo per valutare appieno le prestazioni del modello. Supponiamo che il modello non riesca a diagnosticare accuratamente condizioni gravi che, sebbene rare, possono avere gravi conseguenze; la sua elevata precisione potrebbe non essere così significativa. Un esempio pertinente di ciò è il modello di apprendimento automatico della retinopatia di Google, progettato per diagnosticare la retinopatia diabetica e l'edema maculare diabetico da fotografie della retina.
Il modello di Google ha dimostrato livelli di precisione impressionanti in contesti di laboratorio. Tuttavia, quando è stato distribuito in ambienti clinici reali in Thailandia, ha dovuto affrontare sfide significative. Nel contesto reale, il modello ha incontrato popolazioni di pazienti diverse, qualità delle immagini variabili e una gamma di diverse condizioni mediche a cui non era stato esposto durante il suo training. Di conseguenza, le sue prestazioni avrebbero potuto essere migliori e ha fatto fatica a mantenere gli stessi livelli di accuratezza osservati in laboratorio. Questo esempio serve come un chiaro promemoria del fatto che, sebbene un'elevata accuratezza sia un attributo importante e desiderabile per un modello di diagnosi medica, deve essere valutata insieme ad altri fattori, come la capacità del modello di generalizzare a diverse popolazioni e gestire condizioni reali diverse e imprevedibili, per comprenderne veramente il valore e il potenziale impatto sull'assistenza ai pazienti.

Allo stesso modo, se il modello funziona bene in media ma mostra significative disparità nelle prestazioni tra diversi gruppi demografici, anche questo sarebbe motivo di preoccupazione.
L'evoluzione dell'apprendimento automatico ha quindi visto uno spostamento verso un approccio più olistico alla valutazione del modello, tenendo conto non solo dell'accuratezza, ma anche di altri fattori cruciali come la correttezza, trasparenza e applicabilità nel mondo reale. Un esempio lampante è il progetto Gender Shades del MIT Media Lab, guidato da Joy Buolamwini, che evidenzia significativi pregiudizi razziali e di genere nei sistemi commerciali di riconoscimento facciale. Il progetto ha valutato le prestazioni di tre tecnologie di riconoscimento facciale sviluppate da IBM, Microsoft e Face++. Ha scoperto che tutte presentavano dei pregiudizi, con prestazioni migliori su volti maschili e dalla pelle più chiara rispetto a volti femminili e dalla pelle più scura.
Sebbene l'accuratezza rimanga una metrica fondamentale e preziosa per la valutazione dei modelli di apprendimento automatico, è necessario un approccio più completo per valutare appieno le prestazioni di un modello. Ciò significa considerare metriche aggiuntive che tengano conto di correttezza, trasparenza e applicabilità nel mondo reale, nonché condurre test rigorosi su diversi set di dati per scoprire e mitigare eventuali potenziali pregiudizi. Il passaggio a un approccio più olistico alla valutazione del modello riflette la maturazione del campo e il suo crescente riconoscimento delle implicazioni nel mondo reale e delle considerazioni etiche associate all'implementazione di modelli di apprendimento automatico.

11.5.2.2 Equità

La correttezza nei modelli di apprendimento automatico è un aspetto multiforme e critico che richiede un'attenzione particolare, in particolare nelle applicazioni ad alto rischio che influenzano significativamente la vita delle persone, come nei processi di approvazione dei prestiti, nelle assunzioni e nella giustizia penale. Si riferisce al trattamento equo di tutti gli individui, indipendentemente dai loro attributi demografici o sociali come razza, genere, età o stato socioeconomico.
Affidarsi semplicemente all'accuratezza può essere insufficiente e potenzialmente fuorviante quando si valutano i modelli. Ad esempio, si consideri un modello di approvazione dei prestiti con un tasso di accuratezza del 95%. Sebbene questa cifra possa sembrare impressionante a prima vista, non rivela come il modello si comporta nei diversi gruppi demografici. Se questo modello discrimina costantemente un gruppo particolare, la sua accuratezza è meno encomiabile e la sua correttezza viene messa in discussione.
La discriminazione può manifestarsi in varie forme, come la discriminazione diretta, in cui un modello utilizza esplicitamente attributi sensibili come razza o genere nel suo processo decisionale, o discriminazione indiretta, in cui variabili apparentemente neutre sono correlate ad attributi sensibili, influenzando indirettamente i risultati del modello. Un esempio infame di quest'ultimo è lo strumento COMPAS utilizzato nel sistema di giustizia penale degli Stati Uniti, che ha mostrato pregiudizi razziali nel prevedere i tassi di recidiva nonostante non utilizzasse esplicitamente la razza come variabile.
Affrontare l'equità implica un attento esame delle prestazioni del modello tra gruppi diversi, identificando potenziali pregiudizi e rettificando le disparità attraverso misure correttive come il ribilanciamento dei set di dati, l'adeguamento dei parametri del modello e l'implementazione di algoritmi consapevoli dell'equità e della correttezza. Ricercatori e professionisti sviluppano continuamente metriche e metodologie su misura per casi d'uso specifici per valutare la correttezza e l'equità in scenari del mondo reale. Ad esempio, l'analisi di impatto disparato, la parità demografica e le pari opportunità sono alcune delle metriche impiegate per valutare l'equità/correttezza.

Inoltre, la trasparenza e l'interpretabilità dei modelli sono fondamentali per raggiungere la correttezza. Comprendere come un modello prende decisioni può rivelare potenziali pregiudizi e consentire alle parti interessate di ritenere responsabili gli sviluppatori. Strumenti open source come AI Fairness 360 di IBM e Fairness Indicators di TensorFlow sono in fase di sviluppo per facilitare le valutazioni dell'equità/correttezza e l'attenuazione dei pregiudizi nei modelli di apprendimento automatico.
Garantire l'equità/correttezza nei modelli di apprendimento automatico, in particolare nelle applicazioni che hanno un impatto significativo sulla vita delle persone, richiede una rigorosa valutazione delle prestazioni del modello in gruppi diversi, un'attenta identificazione e attenuazione dei pregiudizi e l'implementazione di misure di trasparenza e interpretabilità. Affrontando la correttezza in modo completo, possiamo lavorare per sviluppare modelli di apprendimento automatico equi, giusti e vantaggiosi per la società.

11.5.2.3 Complessità

11.5.2.3.1 Parametri

Nelle fasi iniziali del machine learning, il benchmarking dei modelli si basava spesso sui conteggi dei parametri come proxy [sostituto] per la complessità del modello. La logica era che più parametri in genere portano a un modello più complesso, che dovrebbe, a sua volta, fornire prestazioni migliori. Tuttavia, questo approccio si è dimostrato inadeguato in quanto deve tenere conto del costo computazionale associato all'elaborazione di molti parametri. Ad esempio, GPT-3, sviluppato da OpenAI, è un modello linguistico che vanta ben 175 miliardi di parametri. Sebbene raggiunga prestazioni all'avanguardia in varie attività di elaborazione del linguaggio naturale, le sue dimensioni e le risorse computazionali necessarie per eseguirlo lo rendono poco pratico per l'implementazione in molti scenari del mondo reale, in particolare quelli con capacità computazionali limitate.
Affidarsi ai conteggi dei parametri come proxy per la complessità del modello non riesce a considerare anche l'efficienza del modello. Se ottimizzato per l'efficienza, un modello con meno parametri potrebbe essere altrettanto efficace, se non di più, di un modello con un conteggio di parametri più elevato. Ad esempio, MobileNets, sviluppato da Google, è una famiglia di modelli progettati specificamente per dispositivi mobili ed edge. Utilizzano convoluzioni separabili in base alla profondità per ridurre il numero di parametri e i costi computazionali, pur mantenendo prestazioni competitive.
Alla luce di queste limitazioni, il settore si è spostato verso un approccio più olistico al benchmarking dei modelli che considera i conteggi dei parametri e altri fattori cruciali come le operazioni in virgola mobile al secondo (FLOP), il consumo di memoria e la latenza. I FLOP, in particolare, sono emersi come una metrica importante in quanto forniscono una rappresentazione più accurata del carico computazionale imposto da un modello. Questo passaggio a un approccio più completo al benchmarking dei modelli riflette il riconoscimento della necessità di bilanciare prestazioni e praticità, assicurando che i modelli siano efficaci, efficienti e implementabili in scenari reali.

11.5.2.3.2 FLOP

La dimensione di un modello di apprendimento automatico è un aspetto essenziale che influisce direttamente sulla sua usabilità in scenari pratici, soprattutto quando le risorse computazionali sono limitate. Tradizionalmente, il numero di parametri in un modello veniva spesso utilizzato come proxy per le sue dimensioni, con l'ipotesi di base che più parametri si sarebbero tradotti in prestazioni migliori. Tuttavia, questa visione semplicistica non considera il costo computazionale dell'elaborazione di questi parametri. È qui che entra in gioco il concetto di "floating-point operations per second (FLOP)" [operazioni in virgola

mobile al secondo], che fornisce una rappresentazione più accurata del carico computazionale imposto da un modello.
I FLOP misurano il numero di operazioni in virgola mobile eseguite da un modello per generare una previsione. Un modello con molti FLOP richiede risorse computazionali sostanziali per elaborare il vasto numero di operazioni, il che potrebbe renderlo poco pratico per alcune applicazioni. Al contrario, un modello con un conteggio di FLOP inferiore è più leggero e può essere facilmente distribuito in scenari in cui le risorse computazionali sono limitate. Figura 11.8, da (Bianco et al. 2018), mostra la relazione tra la Top-1 Accuracy su ImageNet (asse *y*), i G-FLOP del modello (asse *x*) e il conteggio dei parametri del modello (dimensione del cerchio).

Figura 11.8: Un grafico che raffigura la top-1 Imagenet Accuracy rispetto al conteggio FLOP di un modello insieme al conteggio dei parametri del modello. La figura mostra un compromesso complessivo tra complessità e accuratezza del modello, sebbene alcune architetture del modello siano più efficienti di altre. Fonte: Bianco et al. (2018).

Consideriamo un esempio. BERT—Bidirectional Encoder Representations from Transformers (Devlin et al. 2019)—è un modello di elaborazione del linguaggio naturale molto diffuso, con oltre 340 milioni di parametri, il che lo rende un modello di grandi dimensioni con elevata accuratezza e prestazioni impressionanti in diverse attività. Tuttavia, le dimensioni di BERT, unite al suo elevato numero di FLOP, lo rendono un modello computazionalmente intensivo

che potrebbe non essere adatto per applicazioni in tempo reale o per l'implementazione su dispositivi edge con capacità computazionali limitate.
Alla luce di ciò, c'è stato un crescente interesse nello sviluppo di modelli più piccoli in grado di raggiungere livelli di prestazioni simili alle loro controparti più grandi, pur essendo più efficienti nel carico computazionale. DistilBERT, ad esempio, è una versione più piccola di BERT che mantiene il 97% delle sue prestazioni, pur essendo il 40% più piccola in termini di numero di parametri. La riduzione delle dimensioni si traduce anche in un numero di FLOP inferiore, rendendo DistilBERT una scelta più pratica per scenari con risorse limitate.
In sintesi, mentre il conteggio dei parametri fornisce un'indicazione utile della dimensione del modello, non è una metrica completa in quanto deve considerare il costo computazionale associato all'elaborazione di questi parametri. I FLOP, d'altro canto, offrono una rappresentazione più accurata del carico computazionale di un modello e sono quindi una considerazione essenziale quando si distribuiscono modelli di apprendimento automatico in scenari reali, in particolare quando le risorse computazionali sono limitate. L'evoluzione dal basarsi esclusivamente sul conteggio dei parametri alla considerazione dei FLOP indica una maturazione nel campo, che riflette una maggiore consapevolezza dei vincoli pratici e delle sfide dell'implementazione di modelli di apprendimento automatico in contesti diversi.

11.5.2.3.3 Efficienza

Anche le metriche di efficienza, come il consumo di memoria e la latenza/capacità di elaborazione, hanno acquisito importanza. Queste metriche sono particolarmente cruciali quando si distribuiscono modelli su dispositivi edge o in applicazioni in tempo reale, poiché misurano la velocità con cui un modello può elaborare i dati e la quantità di memoria richiesta. In questo contesto, le curve di Pareto vengono spesso utilizzate per visualizzare il compromesso tra diverse metriche, aiutando le parti interessate a decidere quale modello si adatta meglio alle loro esigenze.

11.5.3 Lezioni Apprese

Il benchmarking dei modelli ci ha offerto diverse preziose intuizioni che possono essere sfruttate per guidare l'innovazione nei benchmark di sistema. La progressione dei modelli di apprendimento automatico è stata profondamente influenzata dall'avvento delle classifiche e dalla disponibilità open source di modelli e set di dati. Questi elementi hanno svolto il ruolo di catalizzatori significativi, spingendo l'innovazione e accelerando l'integrazione di modelli all'avanguardia negli ambienti di produzione. Tuttavia, come approfondiremo ulteriormente, questi non sono gli unici fattori che contribuiscono allo sviluppo dei benchmark di apprendimento automatico.
Le classifiche svolgono un ruolo fondamentale nel fornire un metodo oggettivo e trasparente per ricercatori e professionisti per valutare l'efficacia di diversi modelli, classificandoli in base alle loro prestazioni nei benchmark. Questo sistema promuove un ambiente competitivo, incoraggiando lo sviluppo di modelli che non siano solo accurati ma anche efficienti. L'ImageNet Large Scale Visual Recognition Challenge (ILSVRC) ne è un ottimo esempio, con la sua classifica annuale che contribuisce in modo significativo allo sviluppo di modelli innovativi come AlexNet.
L'accesso open source a modelli e set di dati all'avanguardia diffonde ulteriormente l'apprendimento automatico, facilitando la collaborazione tra ricercatori e professionisti in tutto il mondo. Questo accesso aperto accelera il processo di test, convalida e distribuzione di nuovi modelli in ambienti di produzione, come dimostrato dall'adozione diffusa di modelli come BERT e GPT-3 in varie applicazioni, dall'elaborazione del linguaggio naturale a compiti multimodali più complessi.

Piattaforme di collaborazione della comunità come Kaggle hanno rivoluzionato il settore ospitando competizioni che uniscono data scientist da tutto il mondo per risolvere problemi intricati. Benchmark specifici fungono da paletti per l'innovazione e lo sviluppo di modelli. Inoltre, la disponibilità di set di dati diversi e di alta qualità è fondamentale per l'addestramento e il test dei modelli di apprendimento automatico. Set di dati come ImageNet hanno svolto un ruolo fondamentale nell'evoluzione dei modelli di riconoscimento delle immagini, mentre ampi set di dati di testo hanno facilitato i progressi nei modelli di elaborazione del linguaggio naturale.
Infine, è necessario supportare i contributi di istituti accademici e di ricerca. Il loro ruolo nella pubblicazione di articoli di ricerca, nella condivisione di risultati in conferenze e nella promozione della collaborazione tra varie istituzioni ha contribuito in modo significativo al progresso dei modelli e dei benchmark di apprendimento automatico.

11.5.3.1 Tendenze Emergenti

Man mano che i modelli di apprendimento automatico diventano più sofisticati, lo diventano anche i benchmark necessari per valutarli in modo accurato. Ci sono diversi benchmark e dataset emergenti che stanno guadagnando popolarità grazie alla loro capacità di valutare i modelli in scenari più complessi e realistici:

Dataset Multimodali: Questi set di dati contengono più tipi di dati, come testo, immagini e audio, per rappresentare meglio le situazioni del mondo reale. Un esempio è VQA (Visual Question Answering) (Antol et al. 2015), in cui viene testata la capacità dei modelli di rispondere a domande basate su testo sulle immagini.

Valutazione di Correttezza e Bias: C'è una crescente attenzione alla creazione di benchmark che valutino l'equità/Correttezza e i bias [pregiudizi] dei modelli di apprendimento automatico. Esempi includono il toolkit AI Fairness 360, che offre un set completo di metriche e set di dati per valutare il bias nei modelli.

Generalizzazione Out-of-Distribution: Test di quanto bene i modelli funzionano su dati diversi dalla distribuzione di training originale. Questo valuta la capacità del modello di generalizzare a dati nuovi e inediti. Esempi di benchmark sono Wilds (Koh et al. 2021), RxRx e ANC-Bench.

Robustezza Avversaria: Valutazione delle prestazioni del modello in caso di attacchi avversari o perturbazioni ai dati di input. Questo testa la robustezza del modello. Esempi di benchmark sono ImageNet-A (Hendrycks et al. 2021), ImageNet-C (C. Xie et al. 2020) e CIFAR-10.1.

Prestazioni nel Mondo Reale: Test di modelli su set di dati del mondo reale che corrispondono da vicino alle attività finali anziché solo su set di dati di benchmark predefiniti.

Esempi sono set di dati di imaging medico per attività sanitarie o log di chat di assistenza clienti per sistemi di dialogo.

Efficienza Energetica e di Calcolo: Benchmark che misurano le risorse di calcolo necessarie per ottenere una particolare accuratezza. Questo valuta l'efficienza del modello. Esempi sono MLPerf e Greenbench, già discussi nella sezione Benchmarking dei sistemi.

Interpretabilità e Spiegabilità: Benchmark che valutano quanto sia facile comprendere e spiegare la logica interna e le previsioni di un modello. Esempi di parametri sono la fedeltà ai gradienti di input e la coerenza delle spiegazioni.

11.5.4 Limitazioni e Sfide

Sebbene i benchmark dei modelli siano uno strumento essenziale per valutare i modelli di machine learning, è necessario affrontare diverse limitazioni e sfide per garantire che riflettano accuratamente le prestazioni in scenari reali.

Il dataset non corrisponde a scenari reali: Spesso, i dati utilizzati nei benchmark dei modelli vengono puliti e preelaborati a tal punto che potrebbe essere necessario rappresentare accuratamente i dati che un modello incontrerebbe in applicazioni reali. Questa versione idealizzata dei dati può portare a una sovrastima delle prestazioni di un modello. Nel caso del set di dati ImageNet, le immagini sono ben etichettate e categorizzate. Tuttavia, in uno scenario reale, un modello potrebbe dover gestire immagini sfocate che potrebbero essere meglio illuminate o scattate da angolazioni scomode. Questa discrepanza può influire in modo significativo sulle prestazioni del modello.

Sim2Real Gap: Il Sim2Real Gap si riferisce alla differenza nelle prestazioni di un modello quando si passa da un ambiente simulato a un ambiente reale. Questo gap è spesso osservato nella robotica, dove un robot addestrato in un ambiente simulato ha difficoltà a svolgere compiti nel mondo reale a causa della complessità e dell'imprevedibilità degli ambienti reali. Un robot addestrato a raccogliere oggetti in un ambiente simulato potrebbe aver bisogno di aiuto per svolgere lo stesso compito nel mondo reale perché l'ambiente simulato non rappresenta accuratamente le complessità della fisica, dell'illuminazione e della variabilità degli oggetti del mondo reale.

Sfide nella Creazione di Dataset: La creazione di un set di dati per il benchmarking del modello è un'attività impegnativa che richiede un'attenta considerazione di vari fattori come qualità dei dati, diversità e rappresentazione. Come discusso nella sezione di ingegneria dei dati, garantire che i dati siano puliti, imparziali e rappresentativi dello scenario del mondo reale è fondamentale per l'accuratezza e l'affidabilità del benchmark. Ad esempio, quando si crea un set di dati per un'attività correlata all'assistenza sanitaria, è importante assicurarsi che i dati siano rappresentativi dell'intera popolazione e non distorti verso un particolare gruppo demografico. Ciò garantisce che il modello funzioni bene in diverse popolazioni di pazienti.

I benchmark del modello sono essenziali per misurare la capacità di un'architettura di modello di risolvere un'attività fissa, ma è importante affrontare le limitazioni e le sfide ad essi associate. Ciò include il garantire che il set di dati rappresenti accuratamente scenari del mondo reale, affrontare il divario Sim2Real e superare le sfide della creazione di set di dati imparziali e rappresentativi. Affrontando queste sfide e molte altre, possiamo garantire che i benchmark del modello forniscano una valutazione più accurata e affidabile delle prestazioni di un modello in applicazioni del mondo reale.

Lo Speech Commands dataset e il suo successore MSWC sono benchmark comuni per una delle applicazioni TinyML per eccellenza, l'individuazione delle parole chiave. I comandi vocali stabiliscono metriche di errore di streaming oltre la precisione di classificazione

standard top-1 più pertinenti al caso d'uso di individuazione delle parole chiave. L'utilizzo di metriche pertinenti ai casi è ciò che eleva un dataset a un benchmark del modello.

11.6 Benchmarking dei Dati

Negli ultimi anni, l'intelligenza artificiale si è concentrata sullo sviluppo di modelli di apprendimento automatico sempre più sofisticati, come i grandi modelli linguistici. L'obiettivo è stato quello di creare modelli in grado di prestazioni di livello umano o sovrumane su un'ampia gamma di attività, addestrandoli su enormi set di dati. Questo approccio incentrato sul modello ha prodotto rapidi progressi, con modelli che hanno ottenuto risultati all'avanguardia su molti benchmark consolidati. Figura 11.9 mostra le prestazioni dei sistemi di intelligenza artificiale rispetto alle prestazioni umane (contrassegnate dalla linea orizzontale a 0) in cinque applicazioni: riconoscimento della scrittura a mano, riconoscimento vocale, riconoscimento delle immagini, comprensione della lettura e comprensione del linguaggio. Negli ultimi dieci anni, le prestazioni dell'intelligenza artificiale hanno superato quelle degli esseri umani.

Figura 11.9: IA e prestazioni umane. Fonte: Kiela et al. (2021).

Tuttavia, le crescenti preoccupazioni su questioni come pregiudizi, sicurezza e robustezza persistono anche nei modelli che raggiungono un'elevata accuratezza sui benchmark standard. Inoltre, alcuni set di dati popolari utilizzati per la valutazione dei modelli stanno iniziando a saturarsi, con modelli che raggiungono prestazioni quasi perfette su divisioni di test esistenti (Kiela et al. 2021). Come semplice esempio, ci sono immagini di test nel classico dataset di cifre scritte a mano MNIST che potrebbero sembrare indecifrabili per la maggior parte dei valutatori umani, ma a cui è stata assegnata un'etichetta quando è stato creato il set di dati: i modelli che concordano con quelle etichette potrebbero sembrare esibire prestazioni sovrumane, ma potrebbero invece catturare solo idiosincrasie del processo di etichettatura e acquisizione dalla creazione del set di dati nel 1994. Con lo stesso spirito, i ricercatori di visione artificiale ora chiedono: "Abbiamo finito con ImageNet?" (Beyer et al. 2020). Ciò evidenzia i limiti nell'approccio convenzionale incentrato sul modello di ottimizzazione dell'accuratezza su set di dati fissi tramite innovazioni architettoniche.
Sta emergendo un paradigma alternativo chiamato IA incentrata sui dati. Invece di trattare i dati come statici e concentrarsi strettamente sulle prestazioni del modello, questo approccio riconosce che i modelli sono validi solo quanto i loro dati di training. Quindi, l'enfasi si sposta sulla cura di dataset di alta qualità che riflettano meglio la complessità del mondo reale, sviluppando benchmark di valutazione più informativi e considerando attentamente come i dati vengono campionati, preelaborati e aumentati. L'obiettivo è ottimizzare il comportamento del modello migliorando i dati anziché semplicemente ottimizzando le

metriche su set di dati imperfetti. L'intelligenza artificiale incentrata sui dati esamina e migliora criticamente i dati stessi per produrre un'intelligenza artificiale utile. Ciò riflette un'importante evoluzione nella mentalità, poiché il campo affronta le carenze di un benchmarking ristretto.

Questa sezione esplorerà le principali differenze tra gli approcci all'intelligenza artificiale incentrati sui modelli e sui dati. Questa distinzione ha importanti implicazioni sul modo in cui eseguiamo il benchmarking dei sistemi di intelligenza artificiale. In particolare, vedremo come concentrarsi sulla qualità dei dati e sull'efficienza può migliorare direttamente le prestazioni dell'apprendimento automatico come alternativa all'ottimizzazione delle sole architetture dei modelli. L'approccio incentrato sui dati riconosce che i modelli sono validi solo quanto i loro dati di addestramento. Quindi, migliorare la cura dei dati, i benchmark di valutazione e i processi di gestione dei dati può produrre sistemi di intelligenza artificiale più sicuri, più equi e più robusti. Ripensare il benchmarking per dare priorità ai dati insieme ai modelli rappresenta un'importante evoluzione, poiché il settore si sforza di fornire un impatto affidabile nel mondo reale.

11.6.1 Limitazioni dell'IA Incentrata sul Modello

Nell'era dell'IA incentrata sul modello, una caratteristica importante era lo sviluppo di architetture di modelli complesse. Ricercatori e professionisti hanno dedicato notevoli sforzi alla progettazione di modelli sofisticati e intricati nella ricerca di prestazioni superiori. Ciò ha spesso comportato l'incorporazione di livelli aggiuntivi e la messa a punto di una moltitudine di iperparametri per ottenere miglioramenti nell'accuratezza. Contemporaneamente, c'era una notevole enfasi sullo sfruttamento di algoritmi avanzati. Questi algoritmi, spesso in prima linea nelle ultime ricerche, sono stati impiegati per migliorare le prestazioni dei modelli di IA. L'obiettivo principale di questi algoritmi era ottimizzare il processo di apprendimento dei modelli, estraendo così il massimo delle informazioni dai dati di addestramento.

Sebbene l'approccio incentrato sul modello sia stato centrale per molti progressi nell'IA, ha diverse aree di miglioramento. Innanzitutto, lo sviluppo di architetture di modelli complesse può spesso portare a un overfitting. Questo è quando il modello funziona bene sui dati di addestramento ma deve generalizzare a nuovi dati mai visti. I layer aggiuntivi e la complessità possono catturare il rumore nei dati di training come se fosse un pattern reale, danneggiando le prestazioni del modello su nuovi dati.

In secondo luogo, affidarsi ad algoritmi avanzati può a volte oscurare la reale comprensione del funzionamento di un modello. Questi algoritmi spesso agiscono come una scatola nera, rendendo difficile interpretare il modo in cui il modello prende decisioni. Questa mancanza di trasparenza può essere un ostacolo significativo, specialmente in applicazioni critiche come sanità e finanza, dove la comprensione del processo decisionale del modello è fondamentale.

In terzo luogo, l'enfasi sul raggiungimento di risultati all'avanguardia su set di dati di riferimento può a volte essere fuorviante. Questi dataset devono rappresentare in modo più completo le complessità e la variabilità dei dati del mondo reale. Un modello che funziona bene su un set di dati di riferimento potrebbe non essere necessariamente generalizzato bene a dati nuovi e mai visti in un'applicazione del mondo reale. Questa discrepanza può portare a una falsa fiducia nelle capacità del modello e ostacolarne l'applicabilità pratica.

Infine, l'approccio incentrato sul modello spesso si basa su grandi set di dati etichettati per l'addestramento. Tuttavia, ottenere tali set di dati richiede tempo e impegno in molti scenari del mondo reale. Questa dipendenza da grandi dataset limita anche l'applicabilità dell'IA in domini in cui i dati sono scarsi o costosi da etichettare.

Come risultato delle ragioni di cui sopra, e di molte altre, la comunità dell'IA sta passando a un approccio più incentrato sui dati. Invece di concentrarsi solo sull'architettura del modello,

i ricercatori stanno ora dando priorità alla cura di set di dati di alta qualità, allo sviluppo di migliori benchmark di valutazione e alla considerazione di come i dati vengono campionati e preelaborati. L'idea chiave è che i modelli sono validi solo quanto i loro dati di training. Quindi, concentrandoci sull'ottenimento dei dati giusti, potremo sviluppare sistemi di intelligenza artificiale più equi, sicuri e allineati con i valori umani. Questo cambiamento incentrato sui dati rappresenta un importante cambiamento di mentalità man mano che l'intelligenza artificiale progredisce.

11.6.2 Verso un'Intelligenza Artificiale Incentrata sui Dati

L'intelligenza artificiale incentrata sui dati è un paradigma che sottolinea l'importanza di dataset di alta qualità, ben etichettati e diversificati nello sviluppo di modelli di intelligenza artificiale. Contrariamente all'approccio incentrato sul modello, che si concentra sulla rifinitura e l'iterazione dell'architettura e dell'algoritmo del modello per migliorare le prestazioni, l'intelligenza artificiale incentrata sui dati dà priorità alla qualità dei dati di input come motore principale per migliorare le prestazioni del modello. I dati di alta qualità sono puliti, ben etichettati e rappresentativi degli scenari del mondo reale che il modello incontrerà. Al contrario, i dati di bassa qualità possono portare a scarse prestazioni del modello, indipendentemente dalla complessità o dalla sofisticatezza dell'architettura del modello.

L'intelligenza artificiale incentrata sui dati pone una forte enfasi sulla pulizia e l'etichettatura dei dati. La pulizia comporta la rimozione di valori anomali, la gestione dei valori mancanti e la risoluzione di altre incongruenze nei dati. L'etichettatura, d'altro canto, comporta l'assegnazione di etichette significative e accurate ai dati. Entrambi questi processi sono fondamentali per garantire che il modello di intelligenza artificiale venga addestrato su dati accurati e pertinenti. Un altro aspetto importante dell'approccio incentrato sui dati è il "data augmentation" [l'aumento dei dati]. Ciò comporta l'aumento artificiale delle dimensioni e della diversità del set di dati applicando varie trasformazioni ai dati, come rotazione, ridimensionamento e capovolgimento delle immagini di addestramento. L'aumento dei dati aiuta a migliorare la robustezza del modello e le capacità di generalizzazione.

Ci sono diversi vantaggi nell'adottare un approccio incentrato sui dati per lo sviluppo dell'intelligenza artificiale. Innanzitutto, porta a prestazioni del modello migliorate e capacità di generalizzazione. Assicurandosi che il modello venga addestrato su dati diversi e di alta qualità, il modello può generalizzare meglio a dati nuovi e mai visti (Mattson et al. 2020b). Inoltre, un approccio incentrato sui dati può spesso portare a modelli più semplici che sono più facili da interpretare e gestire. Questo perché l'enfasi è sui dati piuttosto che sull'architettura del modello, il che significa che i modelli più semplici possono raggiungere prestazioni elevate quando addestrati su dati di alta qualità.

Il passaggio all'IA incentrata sui dati rappresenta un significativo cambiamento di paradigma. Dando priorità alla qualità dei dati di input, questo approccio cerca di modellare le prestazioni e le capacità di generalizzazione, portando in ultima analisi a sistemi di intelligenza artificiale più solidi e affidabili. Figura 11.10 illustra questa differenza. Mentre continuiamo ad avanzare nella nostra comprensione e applicazione dell'IA, è probabile che l'approccio incentrato sui dati svolga un ruolo importante nel plasmare il futuro di questo campo.

Figura 11.10: Sviluppo di machine learning incentrato sul modello e incentrato sui dati. Fonte: NVIDIA

11.6.3 Benchmarking dei Dati

Il benchmarking dei dati mira a valutare problemi comuni nei set di dati, come l'identificazione di errori di etichetta, caratteristiche rumorose, squilibrio di rappresentazione (ad esempio, su 1000 classi in Imagenet-1K, ci sono oltre 100 categorie che sono solo tipi di cani), squilibrio di classe (dove alcune classi hanno molti più campioni di altre), se i modelli addestrati su un dato set di dati possono generalizzare a caratteristiche fuori distribuzione o quali tipi di bias potrebbero esistere in un dato set di dati (Mattson et al. 2020b). Nella sua forma più semplice, il benchmarking dei dati mira a migliorare l'accuratezza su un set di test rimuovendo campioni di addestramento rumorosi o etichettati in modo errato mantenendo fissa l'architettura del modello. Recenti competizioni nel benchmarking dei dati hanno invitato i partecipanti a presentare nuove strategie di "augmentation" e tecniche di apprendimento attivo.

Le tecniche incentrate sui dati continuano a guadagnare attenzione nel benchmarking, soprattutto perché i modelli di base sono sempre più addestrati su obiettivi auto-supervisionati. Rispetto ai set di dati più piccoli come Imagenet-1K, i set di dati più grandi comunemente usati nell'apprendimento auto-supervisionato, come Common Crawl, OpenImages e LAION-5B, contengono quantità maggiori di rumore, duplicati, bias e dati potenzialmente offensivi.

DataComp è una competizione di dataset lanciata di recente che ha come obiettivo la valutazione di grandi corpora. DataComp si concentra sulle coppie linguaggio-immagine usate per addestrare i modelli CLIP. Il documento introduttivo rileva che quando il budget di elaborazione totale per l'addestramento è costante, i modelli CLIP più performanti nelle attività downstream, come la classificazione ImageNet, vengono addestrati solo sul 30% del pool di campioni disponibile. Ciò suggerisce che un corretto filtraggio di grandi corpora è fondamentale per migliorare l'accuratezza dei modelli di base. Analogamente, Demystifying CLIP Data (H. Xu et al. 2023) chiede se il successo di CLIP sia attribuibile all'architettura o al set di dati.

DataPerf è un altro recente lavoro incentrato sul benchmarking dei dati in varie modalità. DataPerf offre round di competizione online per stimolare il miglioramento dei dataset.

L'offerta inaugurale è stata lanciata con sfide in termini di visione, parlato, acquisizione, debug e prompt di testo per la generazione di immagini.

11.6.4 Efficienza dei Dati

Man mano che i modelli di apprendimento automatico diventano più grandi e complessi e le risorse di elaborazione diventano più scarse di fronte alla crescente domanda, diventa difficile soddisfare i requisiti di elaborazione anche con le flotte di machine learning più grandi. Per superare queste sfide e garantire la scalabilità del sistema di apprendimento automatico, è necessario esplorare nuove opportunità che aumentino gli approcci convenzionali alla scalabilità delle risorse.

Migliorare la qualità dei dati può essere un metodo utile per avere un impatto significativo sulle prestazioni del sistema di apprendimento automatico. Uno dei principali vantaggi del miglioramento della qualità dei dati è il potenziale di poter ridurre le dimensioni del set di dati di addestramento mantenendo o addirittura migliorando le prestazioni del modello. Questa riduzione delle dimensioni dei dati è direttamente correlata alla quantità di tempo di addestramento richiesto, consentendo così ai modelli di convergere in modo più rapido ed efficiente. Raggiungere questo equilibrio tra qualità dei dati e dimensioni del set di dati è un compito impegnativo che richiede lo sviluppo di metodi, algoritmi e tecniche sofisticati.

Possono essere adottati diversi approcci per migliorare la qualità dei dati. Questi metodi includono e non sono limitati a quanto segue:

- **Pulizia dei Dati:** Ciò comporta la gestione dei valori mancanti, la correzione degli errori e la rimozione dei valori anomali. I dati puliti assicurano che il modello non stia imparando da rumore o imprecisioni.
- **Interpretabilità e Spiegabilità dei Dati:** Le tecniche comuni includono LIME (Ribeiro, Singh, e Guestrin 2016), che fornisce informazioni sui limiti decisionali dei classificatori, e valori Shapley (Lundberg e Lee 2017), che stimano l'importanza dei singoli campioni nel contribuire alle previsioni di un modello.
- **Feature Engineering:** Trasformare o creare nuove funzionalità può migliorare significativamente le prestazioni del modello fornendo informazioni più pertinenti per l'apprendimento.
- **Data Augmentation:** Aumentare i dati creando nuovi campioni tramite varie trasformazioni può aiutare a migliorare la robustezza e la generalizzazione del modello.
- **Active Learning:** Questo è un approccio di apprendimento semi-supervisionato in cui il modello interroga attivamente un "oracolo" umano per etichettare i campioni più informativi (Coleman et al. 2022). Ciò garantisce che il modello venga addestrato sui dati più rilevanti.
- **Riduzione della Dimensionalità:** Tecniche come PCA possono ridurre il numero di feature in un set di dati, riducendo così la complessità e il tempo di training.

Esistono molti altri metodi in circolazione. Ma l'obiettivo è lo stesso. Affinare il set di dati e garantire che sia della massima qualità può ridurre il tempo di addestramento necessario per la convergenza dei modelli. Tuttavia, per raggiungere questo obiettivo è necessario sviluppare e implementare metodi, algoritmi e tecniche sofisticati in grado di pulire, preelaborare e aumentare i dati, mantenendo al contempo i campioni più informativi. Questa è una sfida continua che richiederà una continua ricerca e innovazione nel campo dell'apprendimento automatico.

11.7 La Tripletta

Mentre i benchmark di sistema, modello e dati sono stati tradizionalmente studiati in modo isolato, si sta diffondendo la consapevolezza che per comprendere e far progredire completamente l'IA, dobbiamo adottare una visione più olistica. Iterando tra sistemi di benchmarking, modelli e dataset insieme, potrebbero emergere nuove intuizioni che non sono evidenti quando questi componenti vengono analizzati separatamente. Le prestazioni del sistema influiscono sulla precisione del modello, le capacità del modello determinano le esigenze dei dati e le caratteristiche dei dati determinano i requisiti del sistema.
Il benchmarking della triade di sistema, modello e dati in modo integrato porterà probabilmente a scoperte sulla progettazione congiunta dei sistemi di IA, sulle proprietà di generalizzazione dei modelli e sul ruolo della cura e della qualità dei dati nel consentire le prestazioni. Piuttosto che benchmark ristretti di singoli componenti, il futuro dell'IA richiede benchmark che valutino la relazione simbiotica tra piattaforme di elaborazione, algoritmi e dati di training. Questa prospettiva a livello di sistema sarà fondamentale per superare le attuali limitazioni e sbloccare il prossimo livello di capacità dell'IA.
Figura 11.11 illustra i molti modi potenziali per far interagire tra loro il benchmarking dei dati, quello dei modelli e quello dell'infrastruttura di sistema. L'esplorazione di queste complesse interazioni probabilmente porterà alla scoperta di nuove opportunità di ottimizzazione e capacità di miglioramento. La tripletta di benchmark di dati, modelli e sistemi offre un ricco spazio per la progettazione congiunta e la co-ottimizzazione.

Figura 11.11: La tripletta del Benchmarking.

Sebbene questa prospettiva integrata rappresenti una tendenza emergente, il settore ha ancora molto da scoprire sulle sinergie e i compromessi tra questi componenti. Mentre eseguiamo il benchmarking iterativo di combinazioni di dati, modelli e sistemi, emergeranno nuove intuizioni che rimangono nascoste quando questi elementi vengono studiati separatamente. Questo approccio di benchmarking multiforme che traccia le intersezioni di dati, algoritmi e hardware promette di essere una strada fruttuosa per importanti progressi nell'intelligenza artificiale, anche se è ancora nelle sue fasi iniziali.

11.8 Benchmark per Tecnologie Emergenti

Date le loro significative differenze rispetto alle tecniche esistenti, le tecnologie emergenti possono essere particolarmente difficili da progettare per i benchmark. I benchmark standard utilizzati per le tecnologie esistenti potrebbero non evidenziare le caratteristiche chiave del nuovo approccio. Al contrario, i nuovi benchmark potrebbero essere visti come artificiosi per favorire la tecnologia emergente rispetto ad altre. Potrebbero essere così diversi dai benchmark esistenti da non poter essere compresi e perdere significato. Pertanto, i benchmark per le tecnologie emergenti devono bilanciare equità, applicabilità e facilità di confronto con quelli esistenti.

Un esempio di tecnologia emergente in cui il benchmarking si è dimostrato particolarmente difficile è nel Neuromorphic Computing. Utilizzando il cervello come fonte di ispirazione per un'intelligenza generale scalabile, robusta ed efficiente dal punto di vista energetico, il calcolo neuromorfico (Schuman et al. 2022) incorpora direttamente meccanismi biologicamente realistici sia negli algoritmi di calcolo che nell'hardware, come le reti neurali spiking (Maass 1997) e le architetture non-von Neumann architectures per eseguirle (M. Davies et al. 2018; Modha et al. 2023). Da una prospettiva full-stack di modelli, tecniche di training e sistemi hardware, il calcolo neuromorfico differisce dall'hardware e dall'intelligenza artificiale convenzionali. Pertanto, esiste una sfida fondamentale nello sviluppo di benchmark equi e utili per guidare la tecnologia.

Un'iniziativa in corso per sviluppare benchmark neuromorfici standard è NeuroBench (Yik et al. 2023). Per un benchmarking adeguato del neuromorfico, NeuroBench segue principi di alto livello di *inclusività* attraverso l'applicabilità di attività e metriche sia alle soluzioni neuromorfiche che non neuromorfiche, *attuabilità* dell'implementazione utilizzando strumenti comuni e aggiornamenti *iterativi* per continuare a garantire la pertinenza man mano che il campo cresce rapidamente. NeuroBench e altri benchmark per le tecnologie emergenti forniscono una guida critica per le tecniche future, che potrebbero essere necessarie man mano che i limiti di scalabilità degli approcci esistenti si avvicinano.

11.9 Conclusione

Ciò che viene misurato viene migliorato. Questo capitolo ha esplorato la natura multiforme del benchmarking che abbraccia sistemi, modelli e dati. Il benchmarking è importante per far progredire l'IA in quanto fornisce le misurazioni essenziali per monitorare i progressi.

I benchmark del sistema ML consentono l'ottimizzazione attraverso metriche di velocità, efficienza e scalabilità. I benchmark del modello guidano l'innovazione attraverso attività e metriche standardizzate oltre l'accuratezza. I benchmark dei dati evidenziano problemi di qualità, equilibrio e rappresentazione.

È importante notare che la valutazione di questi componenti in modo isolato presenta dei limiti. In futuro, sarà probabilmente utilizzato un benchmarking più integrato per esplorare l'interazione tra benchmark di sistema, modello e dati. Questa visione promette nuove intuizioni sulla progettazione congiunta di dati, algoritmi e infrastrutture.

Man mano che l'IA diventa più complessa, il benchmarking completo diventa ancora più critico. Gli standard devono evolversi continuamente per misurare nuove capacità e rivelare limitazioni. Una stretta collaborazione tra settore, mondo accademico, etichette nazionali, ecc. è essenziale per sviluppare benchmark rigorosi, trasparenti e socialmente utili.

Il benchmarking fornisce la bussola per guidare il progresso nell'IA. Misurando costantemente e condividendo apertamente i risultati, possiamo orientarci verso sistemi performanti, robusti e affidabili. Se l'IA deve soddisfare adeguatamente le esigenze sociali e umane, deve essere sottoposta a benchmarking tenendo a mente gli interessi dell'umanità. A tal fine, ci sono aree emergenti, come il benchmarking della sicurezza dei sistemi di IA, ma

questo è per un altro giorno e qualcosa di cui possiamo discutere ulteriormente in "Generative AI"!
Il benchmarking è un argomento in continua evoluzione. L'articolo The Olympics of AI: Benchmarking Machine Learning Systems copre diversi sottocampi emergenti nel benchmarking dell'IA, tra cui robotica, realtà estesa e calcolo neuromorfico che incoraggiamo il lettore ad approfondire.

11.10 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Perché il benchmarking è importante?
- Benchmarking di inferenza embedded.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 11.1
- Esercizio 11.2

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

12. Apprendimento On-Device

DALL·E 3 Prompt: Disegno di uno smartphone con i suoi componenti interni esposti, che mostra diversi ingegneri in miniatura di diversi sessi e tonalità di pelle che lavorano attivamente sul modello di ML. Gli ingegneri, tra cui uomini, donne e individui non binari, stanno regolando i parametri, riparando le connessioni e migliorando la rete al volo. I dati fluiscono nel modello ML, vengono elaborati in tempo reale e generano inferenze di output.

L'apprendimento "On-device" [sul dispositivo] rappresenta un'innovazione significativa per i dispositivi IoT embedded ed edge, consentendo ai modelli di addestrarsi e aggiornarsi direttamente su piccoli dispositivi locali. Ciò contrasta con i metodi tradizionali, in cui i modelli vengono addestrati su ampie risorse di cloud computing prima della distribuzione. Con l'apprendimento On-Device, dispositivi come smart speaker, dispositivi indossabili e sensori industriali possono perfezionare i modelli in tempo reale in base ai dati locali senza dover trasmettere dati esternamente. Ad esempio, uno smart speaker con comando vocale potrebbe apprendere e adattarsi ai pattern di linguaggio e al vocabolario del suo proprietario direttamente sul dispositivo. Tuttavia, non esiste un "pranzo gratis"; pertanto, in questo capitolo, discuteremo sia i vantaggi che i limiti dell'apprendimento sul dispositivo.

Obiettivi dell'Apprendimento

- Comprendere l'apprendimento sul dispositivo e in che modo differisce dal training basato su cloud
- Riconoscere i vantaggi e i limiti dell'apprendimento sul dispositivo
- Esaminare le strategie per adattare i modelli tramite riduzione della complessità, ottimizzazione e compressione dei dati
- Comprendere concetti correlati come apprendimento federato e apprendimento tramite trasferimento
- Analizzare le implicazioni della sicurezza dell'apprendimento sul dispositivo e delle strategie di mitigazione

12.1 Introduzione

L'apprendimento su dispositivo si riferisce all'addestramento di modelli ML direttamente sul dispositivo in cui vengono distribuiti, al contrario dei metodi tradizionali in cui i modelli

vengono addestrati su server potenti e poi distribuiti sui dispositivi. Questo metodo è particolarmente rilevante per TinyML, in cui i sistemi ML sono integrati in dispositivi minuscoli e con risorse limitate.

Un esempio di apprendimento su dispositivo può essere visto in un termostato intelligente che si adatta al comportamento dell'utente nel tempo. Inizialmente, il termostato può avere un modello generico che comprende pattern di utilizzo di base. Tuttavia, poiché è esposto a più dati, come gli orari in cui l'utente è a casa o fuori, le temperature preferite e le condizioni meteorologiche esterne, il termostato può perfezionare il suo modello direttamente sul dispositivo per fornire un'esperienza personalizzata. Tutto ciò avviene senza inviare dati a un server centrale per l'elaborazione.

Un altro esempio è nel testo predittivo sugli smartphone. Mentre gli utenti digitano, il telefono impara dai pattern linguistici dell'utente e suggerisce parole o frasi che probabilmente verranno utilizzate in seguito. Questo apprendimento avviene direttamente sul dispositivo e il modello si aggiorna in tempo reale man mano che vengono raccolti più dati. Un esempio pratico di apprendimento su dispositivo ampiamente utilizzato è Gboard. Su un telefono Android, Gboard impara da modelli di digitazione e dettatura per migliorare l'esperienza per tutti gli utenti. L'apprendimento "On-device" è anche chiamato "apprendimento federato". Figura 12.1 mostra il ciclo di apprendimento federato sui dispositivi mobili: *A.* il dispositivo impara dai pattern utente; *B.* gli aggiornamenti del modello locale vengono comunicati al cloud; *C.* il server cloud aggiorna il modello globale e invia il nuovo modello a tutti i dispositivi.

Figura 12.1: Ciclo di apprendimento federato. Fonte: Google Research.

12.2 Vantaggi e Limiti

L'apprendimento su dispositivo offre diversi vantaggi rispetto al tradizionale ML basato su cloud. Mantenendo dati e modelli sul dispositivo, elimina la necessità di costose trasmissioni di dati e risolve i problemi di privacy. Ciò consente esperienze più personalizzate e reattive, poiché il modello può adattarsi in tempo reale al comportamento dell'utente.

Tuttavia, l'apprendimento su dispositivo presenta anche dei compromessi. Le limitate risorse di elaborazione sui dispositivi dei consumatori possono rendere difficile l'esecuzione di modelli complessi in locale. Anche i set di dati sono più limitati poiché sono costituiti solo da

dati generati dall'utente da un singolo dispositivo. Inoltre, l'aggiornamento dei modelli richiede l'invio di nuove versioni anziché aggiornamenti cloud senza interruzioni.
L'apprendimento su dispositivo apre nuove possibilità abilitando l'intelligenza artificiale offline mantenendo al contempo la privacy dell'utente. Tuttavia, richiede una gestione attenta della complessità dei modelli e dei dati entro i limiti dei dispositivi dei consumatori. Trovare il giusto equilibrio tra localizzazione e offload dal cloud è fondamentale per ottimizzare le esperienze su dispositivo.

12.2.1 Vantaggi

12.2.1.1 Privacy e Sicurezza dei Dati

Uno dei vantaggi significativi dell'apprendimento sul dispositivo è la maggiore privacy e sicurezza dei dati degli utenti. Ad esempio, si consideri uno smartwatch che monitora parametri sanitari sensibili come la frequenza cardiaca e la pressione sanguigna. Elaborando i dati e adattando i modelli direttamente sul dispositivo, i dati biometrici rimangono localizzati, aggirando la necessità di trasmettere dati grezzi ai server cloud dove potrebbero essere soggetti a violazioni.
Le violazioni dei server sono tutt'altro che rare, con milioni di record compromessi ogni anno. Ad esempio, la violazione di Equifax del 2017 ha esposto i dati personali di 147 milioni di persone. Mantenendo i dati sul dispositivo, il rischio di tali esposizioni è drasticamente ridotto. L'apprendimento sul dispositivo elimina la dipendenza dall'archiviazione cloud centralizzata e protegge dall'accesso non autorizzato da varie minacce, tra cui attori malintenzionati, minacce interne ed esposizione accidentale.
Regolamenti come l'Health Insurance Portability and Accountability Act (HIPAA) e il General Data Protection Regulation (GDPR) impongono rigorosi requisiti di riservatezza dei dati che l'apprendimento sul dispositivo affronta abilmente. Garantendo che i dati rimangano localizzati e non vengano trasferiti ad altri sistemi, l'apprendimento sul dispositivo facilita la conformità a tali regolamenti.
L'apprendimento sul dispositivo non è solo vantaggioso per i singoli utenti; ha implicazioni significative per le organizzazioni e i settori che gestiscono dati altamente sensibili. Ad esempio, in ambito militare, l'apprendimento sul dispositivo consente ai sistemi di prima linea di adattare modelli e funzioni indipendentemente dalle connessioni ai server centrali che potrebbero essere potenzialmente compromessi. Le informazioni critiche e sensibili sono saldamente protette dalla localizzazione dell'elaborazione e dell'apprendimento dei dati. Tuttavia, ciò comporta il compromesso che i singoli dispositivi assumono più valore e possono incentivare furti o distruzioni poiché diventano gli unici vettori di modelli di intelligenza artificiale specializzati. È necessario prestare attenzione alla protezione dei dispositivi stessi durante la transizione all'apprendimento sul dispositivo.
È inoltre importante preservare la privacy, la sicurezza e la conformità normativa dei dati personali e sensibili. Invece che nel cloud, i modelli di training e operativi aumentano sostanzialmente le misure di privacy a livello locale, assicurando che i dati degli utenti siano protetti da potenziali minacce.
Tuttavia, questo è solo parzialmente intuitivo perché l'apprendimento sul dispositivo potrebbe invece esporre i sistemi a nuovi attacchi alla privacy. Con preziosi riepiloghi dei dati e aggiornamenti dei modelli archiviati in modo permanente su singoli dispositivi, potrebbe essere molto più difficile proteggerli fisicamente e digitalmente rispetto a un grande cluster di elaborazione. Mentre l'apprendimento sul dispositivo riduce la quantità di dati compromessi in una qualsiasi violazione, potrebbe anche introdurre nuovi pericoli disperdendo informazioni sensibili su molti terminali decentralizzati. Le pratiche di sicurezza attente sono ancora essenziali per i sistemi "on-device".

12.2.1.2 Normativa di Conformità

L'apprendimento sul dispositivo aiuta ad affrontare le principali normative sulla privacy come GDPR)e CCPA. Queste normative richiedono la localizzazione dei dati, limitando i trasferimenti di dati transfrontalieri a paesi approvati con controlli adeguati. Il GDPR impone inoltre requisiti di "privacy by design" e consenso per la raccolta dei dati. Mantenendo l'elaborazione dei dati e il training del modello localizzati sul dispositivo, i dati sensibili degli utenti non vengono trasferiti altrove. Ciò evita importanti grattacapi di conformità per le organizzazioni.

Ad esempio, un fornitore di servizi sanitari che monitora i parametri vitali dei pazienti con dispositivi indossabili deve garantire che i trasferimenti di dati transfrontalieri siano conformi a HIPAA e GDPR se utilizza il cloud. Determinare le leggi del paese applicabili e garantire le approvazioni per i flussi di dati internazionali introduce oneri legali e ingegneristici. Con l'apprendimento on-device, nessun dato lascia il dispositivo, semplificando la conformità. Il tempo e le risorse spesi per la conformità vengono ridotti in modo significativo.

Settori come sanità, finanza e governo, che hanno dati altamente regolamentati, possono trarre grandi vantaggi dal training sul dispositivo. Localizzando i dati e l'apprendimento, i requisiti normativi di privacy e sovranità dei dati vengono soddisfatti più facilmente. Le soluzioni su dispositivo forniscono un modo efficiente per creare applicazioni di IA conformi.

Le principali normative sulla privacy impongono restrizioni sullo spostamento transfrontaliero dei dati che l'apprendimento su dispositivo affronta intrinsecamente tramite elaborazione localizzata. Ciò riduce l'onere di conformità per le organizzazioni che lavorano con dati regolamentati.

12.2.1.3 Riduzione della Larghezza di Banda, dei Costi e Maggiore Efficienza

Uno dei principali vantaggi dell'apprendimento su dispositivo è la significativa riduzione dell'utilizzo della larghezza di banda e dei costi associati all'infrastruttura cloud. Mantenendo i dati localizzati per l'addestramento del modello anziché trasmettere dati grezzi al cloud, l'apprendimento su dispositivo può comportare notevoli risparmi di larghezza di banda. Ad esempio, una rete di telecamere che analizzano i filmati video può ottenere significative riduzioni nel trasferimento di dati addestrando i modelli sul dispositivo anziché trasmettere in streaming tutti i filmati video al cloud per l'elaborazione.

Questa riduzione nella trasmissione dei dati consente di risparmiare larghezza di banda e si traduce in costi inferiori per server, reti e archiviazione dei dati nel cloud. Le grandi organizzazioni, che potrebbero spendere milioni in infrastrutture cloud per addestrare modelli di dati sui dispositivi, possono sperimentare drastiche riduzioni dei costi tramite l'apprendimento on-device. Nell'era dell'intelligenza artificiale generativa, in cui i costi sono aumentati in modo significativo, trovare modi per contenere le spese è diventato sempre più importante.

Inoltre, anche i costi energetici e ambientali della gestione di grandi server farm sono diminuiti. I data center consumano grandi quantità di energia, contribuendo alle emissioni di gas serra. Riducendo la necessità di un'ampia infrastruttura basata su cloud, l'apprendimento sui dispositivi contribuisce a mitigare l'impatto ambientale dell'elaborazione dei dati (C.-J. Wu et al. 2022).

Specificamente per le applicazioni endpoint [finali], l'apprendimento sui dispositivi riduce al minimo il numero di chiamate API di rete necessarie per eseguire l'inferenza tramite un provider cloud. I costi cumulativi associati alla larghezza di banda e alle chiamate API possono aumentare rapidamente per le applicazioni con milioni di utenti. Al contrario, eseguire training e inferenze localmente è notevolmente più efficiente e conveniente. Con ottimizzazioni all'avanguardia, è stato dimostrato che l'apprendimento on-device riduce i

requisiti di memoria del training, migliora drasticamente l'efficienza della memoria e riduce fino al 20% la latenza per iterazione (Dhar et al. 2021).
Un altro vantaggio fondamentale dell'apprendimento sul dispositivo è la possibilità per i dispositivi IoT di adattare continuamente il loro modello ML a nuovi dati per un apprendimento continuo e permanente. I modelli sul dispositivo possono rapidamente diventare obsoleti man mano che il comportamento dell'utente, i pattern di dati e le preferenze cambiano. L'apprendimento continuo consente al modello di adattarsi in modo efficiente a nuovi dati e miglioramenti e di mantenere elevate prestazioni del modello nel tempo.

12.2.2 Limitazioni

Mentre i tradizionali sistemi ML basati su cloud hanno accesso a risorse di elaborazione pressoché infinite, l'apprendimento sul dispositivo è spesso limitato nella potenza di elaborazione e di archiviazione del dispositivo edge su cui viene addestrato il modello. Per definizione, un dispositivo edge è un dispositivo con risorse di elaborazione, memoria ed energia limitate che non possono essere facilmente aumentate o diminuite. Pertanto, la dipendenza dai dispositivi edge può limitare la complessità, l'efficienza e le dimensioni dei modelli ML sul dispositivo.

12.2.2.1 Risorse di elaborazione

I tradizionali sistemi ML basati su cloud utilizzano grandi server con più GPU o TPU di fascia alta, che forniscono una potenza di calcolo e una memoria pressoché infinite. Ad esempio, servizi come Amazon Web Services (AWS) EC2 consentono di configurare cluster di istanze GPU per un training parallelo massiccio.
Al contrario, l'apprendimento sul dispositivo è limitato dall'hardware del dispositivo edge su cui viene eseguito. I dispositivi edge si riferiscono a endpoint come smartphone, elettronica embedded e dispositivi IoT. Per definizione, questi dispositivi hanno risorse di elaborazione, memoria ed energia molto limitate rispetto al cloud.
Ad esempio, uno smartphone tipico o Raspberry Pi può avere solo pochi core CPU, pochi GB di RAM e una piccola batteria. Ancora più limitati in termini di risorse sono i dispositivi microcontrollore TinyML come Arduino Nano BLE Sense. Le risorse sono fisse su questi dispositivi e non possono essere facilmente aumentate su richiesta, come il ridimensionamento dell'infrastruttura cloud. Questa dipendenza dai dispositivi edge limita direttamente la complessità, l'efficienza e le dimensioni dei modelli che possono essere distribuiti per l'addestramento sul dispositivo:

- **Complessità:** I limiti di memoria, elaborazione e potenza limitano la progettazione dell'architettura del modello, così come il numero di layer e dei parametri.
- **Efficienza:** I modelli devono essere fortemente ottimizzati tramite metodi come la quantizzazione e la potatura per essere eseguiti più velocemente e consumare meno energia.
- **Dimensioni:** I file del modello effettivo devono essere compressi il più possibile per rientrare nei limiti di archiviazione dei dispositivi edge.

Pertanto, mentre il cloud offre una scalabilità infinita, l'apprendimento sul dispositivo deve operare entro i rigidi vincoli di risorse dell'hardware. Ciò richiede un'attenta progettazione congiunta di modelli semplificati, metodi di addestramento e ottimizzazioni su misura specificamente per i dispositivi edge.

12.2.2.2 Dimensioni, Accuratezza e Generalizzazione del Dataset

Oltre alle risorse di elaborazione limitate, l'apprendimento sul dispositivo è anche limitato dal set di dati disponibile per i modelli di training.

Nel cloud, i modelli vengono addestrati su dataset enormi e diversi come ImageNet o Common Crawl. Ad esempio, ImageNet contiene oltre 14 milioni di immagini attentamente categorizzate in migliaia di classi.
L'apprendimento sul dispositivo si basa invece su "data silos" più piccoli e decentralizzati, unici per ogni dispositivo. Il rullino fotografico di uno smartphone potrebbe contenere solo migliaia di foto degli interessi e degli ambienti degli utenti.
Questi dati decentralizzati portano alla necessità di dati IID (indipendenti e distribuiti in modo identico). Ad esempio, due amici potrebbero scattare molte foto degli stessi luoghi e oggetti, il che significa che le loro distribuzioni di dati sono altamente correlate piuttosto che indipendenti.
Motivi per cui i dati potrebbero essere non IID nelle impostazioni sul dispositivo:

- **Eterogeneità degli utenti:** Utenti diversi hanno interessi e ambienti diversi.
- **Differenze tra dispositivi:** Sensori, regioni e dati demografici influenzano i dati.
- **Effetti temporali:** Ora del giorno, impatti stagionali sui dati.

L'efficacia del ML si basa in gran parte su dati di training ampi e diversificati. Con set di dati piccoli e localizzati, i modelli on-device potrebbero non riuscire a generalizzare tra diverse popolazioni di utenti e ambienti. Ad esempio, un modello di rilevamento delle malattie addestrato solo su immagini di un singolo ospedale non si generalizzerebbe bene ad altri dati demografici dei pazienti. Le prestazioni nel mondo reale non potranno che migliorare con progressi medici estesi e diversificati. Quindi, mentre l'apprendimento basato su cloud sfrutta enormi set di dati, l'apprendimento su dispositivo si basa su "silo di dati" decentralizzati molto più piccoli, unici per ogni utente.
I dati limitati e le ottimizzazioni richieste per l'apprendimento on-device possono avere un impatto negativo sulla precisione e sulla generalizzazione del modello:

- I piccoli dataset aumentano il rischio di overfitting. Ad esempio, un classificatore di frutta addestrato su 100 immagini rischia di overfitting rispetto a uno addestrato su 1 milione di immagini diverse.
- I dati rumorosi generati dall'utente riducono la qualità. Il rumore del sensore o l'etichettatura impropria dei dati da parte di non esperti possono degradare l'addestramento.
- Ottimizzazioni come la potatura e la quantizzazione compromettono la precisione per l'efficienza. Un modello quantizzato a 8 bit funziona più velocemente ma meno accuratamente di un modello a 32 bit.

Quindi, mentre i modelli cloud raggiungono un'elevata precisione con enormi set di dati e senza vincoli, i modelli su dispositivo possono avere difficoltà a generalizzare. Alcuni studi dimostrano che il training sul dispositivo corrisponde all'accuratezza del cloud su determinate attività. Tuttavia, le prestazioni sui carichi di lavoro reali richiedono ulteriori studi (J. Lin et al. 2022).
Ad esempio, un modello cloud può rilevare con precisione la polmonite nelle radiografie del torace di migliaia di ospedali. Tuttavia, un modello sul dispositivo addestrato solo su una piccola popolazione locale di pazienti potrebbe non riuscire a generalizzare.
Un'accuratezza inaffidabile limita l'applicabilità nel mondo reale dell'apprendimento sul dispositivo per usi critici come la diagnosi di malattie o i veicoli a guida autonoma.
Il training sul dispositivo è anche più lento del cloud a causa delle risorse limitate. Anche se ogni iterazione è più veloce, il processo di training complessivo richiede più tempo.
Ad esempio, un'applicazione di robotica in tempo reale potrebbe richiedere aggiornamenti del modello entro millisecondi. L'On-device training su un piccolo hardware embedded

potrebbe richiedere secondi o minuti per l'aggiornamento, troppo lento per l'uso in tempo reale.
Le sfide relative a precisione, generalizzazione e velocità pongono ostacoli all'adozione dell'apprendimento on-device per sistemi di produzione reali, soprattutto quando affidabilità e bassa latenza sono fondamentali.

12.3 Adattamento On-device

In un'attività ML, il consumo di risorse proviene principalmente da tre fonti:

- Il modello ML stesso;
- Il processo di ottimizzazione durante l'apprendimento del modello
- Archiviazione ed elaborazione del dataset utilizzato per l'apprendimento.

Di conseguenza, ci sono tre approcci per adattare gli algoritmi ML esistenti su dispositivi con risorse limitate:

- Riduzione della complessità del modello ML
- Modifica delle ottimizzazioni per ridurre i requisiti delle risorse di training
- Creazione di nuove rappresentazioni dei dati più efficienti in termini di archiviazione

Nella sezione seguente, esamineremo questi metodi di adattamento dell'apprendimento on-device. Il capitolo Ottimizzazioni dei Modelli fornisce maggiori dettagli sulle ottimizzazioni del modello.

12.3.1 Riduzione della Complessità del Modello

In questa sezione, discuteremo brevemente i modi per ridurre la complessità del modello quando si adattano i modelli ML sul dispositivo. Per i dettagli sulla riduzione della complessità del modello, fare riferimento al capitolo Ottimizzazioni dei Modelli.

12.3.1.1 Algoritmi ML tradizionali

A causa delle limitazioni di elaborazione e memoria dei dispositivi edge, alcuni algoritmi ML tradizionali sono ottimi candidati per applicazioni di apprendimento on-device grazie alla loro natura leggera. Alcuni esempi di algoritmi con basso impatto sulle risorse includono Naive Bayes Classifiers, Support Vector Machines (SVM), Linear Regression, Logistic Regression e algoritmi Decision Tree selezionati.
Con alcuni perfezionamenti, questi algoritmi ML classici possono essere adattati a specifiche architetture hardware ed eseguire attività semplici. I loro bassi requisiti di prestazioni semplificano l'integrazione dell'apprendimento continuo anche su dispositivi edge.

12.3.1.2 Pruning

Il "pruning" [potatura] è una tecnica per ridurre le dimensioni e la complessità di un modello ML per migliorarne l'efficienza e le prestazioni di generalizzazione. Ciò è utile per l'addestramento di modelli su dispositivi edge, in cui vogliamo ridurre al minimo l'utilizzo delle risorse mantenendo un'accuratezza competitiva.
L'obiettivo principale della potatura è rimuovere parti del modello che non contribuiscono in modo significativo al suo potere predittivo, mantenendo al contempo gli aspetti più informativi. Nel contesto degli alberi decisionali, la potatura comporta la rimozione di alcuni rami (sottoalberi) dall'albero, portando a un albero più piccolo e semplice. Allo stesso modo, quando applicato alle Deep Neural Networks (DNN), il pruning riduce il numero di neuroni (unità) o connessioni nella rete. Questo processo è illustrato in Figura 12.2, che dimostra come il pruning può semplificare la struttura di una rete neurale eliminando connessioni o unità meno importanti, con conseguente creazione di un modello più compatto ed efficiente.

Figura 12.2: Potatura della rete.

12.3.1.3 Riduzione della Complessità dei Modelli di Deep Learning

I framework DNN tradizionali basati su cloud hanno un sovraccarico di memoria troppo elevato per essere utilizzati sul dispositivo. Ad esempio, i sistemi di deep learning come PyTorch e TensorFlow richiedono centinaia di megabyte di overhead di memoria durante l'addestramento di modelli come MobilenetV2 e l'overhead aumenta con l'aumentare del numero di parametri di addestramento.

La ricerca attuale per DNN leggeri esplora principalmente architetture CNN. Esistono anche diversi framework "bare-metal" [tutto in hardware] progettati per eseguire reti neurali su MCU mantenendo bassi l'overhead computazionale e l'ingombro di memoria. Alcuni esempi includono MNN, TVM e TensorFlow Lite. Tuttavia, possono eseguire l'inferenza solo durante i passaggi in avanti e non supportano la backpropagation. Sebbene questi modelli siano progettati per l'implementazione edge, la loro riduzione nei pesi del modello e nelle connessioni architettoniche ha portato a minori requisiti di risorse per l'apprendimento continuo.

Il compromesso tra prestazioni e supporto del modello è chiaro quando si adattano i sistemi DNN più diffusi. Come adattiamo i modelli DNN esistenti a impostazioni con risorse limitate mantenendo il supporto per la backpropagation e l'apprendimento continuo? Le ultime ricerche suggeriscono tecniche di progettazione congiunta di algoritmi e sistemi che aiutano a ridurre il consumo di risorse dell'addestramento ML sui dispositivi edge. Utilizzando tecniche come il "quantization-aware scaling" (QAS) [ridimensionamento consapevole della quantizzazione], aggiornamenti sparsi e altre tecniche all'avanguardia, l'apprendimento sul dispositivo è possibile su sistemi embedded con poche centinaia di kilobyte di RAM senza memoria aggiuntiva mantenendo un'elevata precisione.

12.3.2 Modifica dei Processi di Ottimizzazione

La scelta della giusta strategia di ottimizzazione è importante per l'addestramento DNN su un dispositivo, poiché consente di trovare un buon minimo locale. Poiché l'addestramento avviene su un dispositivo, questa strategia deve anche considerare la memoria e la potenza limitate.

12.3.2.1 Quantization-Aware Scaling

La quantizzazione è un metodo comune per ridurre l'impronta di memoria dell'addestramento DNN. Sebbene ciò possa introdurre nuovi errori, questi possono essere mitigati progettando un modello per caratterizzare questo errore statistico. Ad esempio, i modelli potrebbero utilizzare l'arrotondamento stocastico o introdurre l'errore di quantizzazione negli aggiornamenti del gradiente.

Una tecnica algoritmica specifica è Quantization-Aware Scaling (QAS), che migliora le prestazioni delle reti neurali su hardware a bassa precisione, come dispositivi edge,

dispositivi mobili o sistemi TinyML, regolando i fattori di scala durante il processo di quantizzazione.
Come abbiamo discusso nel capitolo Ottimizzazioni dei Modelli, la quantizzazione è il processo di mappatura di un intervallo continuo di valori in un set discreto di valori. Nel contesto delle reti neurali, questo spesso comporta la riduzione della precisione di pesi e attivazioni da virgola mobile a 32 bit a formati di precisione inferiore come gli interi a 8 bit. Questa riduzione della precisione può ridurre significativamente il costo computazionale e l'ingombro di memoria del modello, rendendolo adatto per l'implementazione su hardware a bassa precisione. Figura 12.3 illustra questo concetto, mostrando un esempio di quantizzazione da float a intero in cui i valori in virgola mobile ad alta precisione vengono mappati in una rappresentazione intera più compatta. Questa rappresentazione visiva aiuta a chiarire come la quantizzazione può mantenere la struttura essenziale dei dati riducendone al contempo la complessità e i requisiti di archiviazione.

Figura 12.3: Quantizzazione float-to-integer. Fonte: Nvidia.

Tuttavia, il processo di quantizzazione può anche introdurre errori di quantizzazione che possono degradare le prestazioni del modello. La scalatura basata sulla quantizzazione è una tecnica che riduce al minimo questi errori regolando i fattori di scala utilizzati nel processo di quantizzazione.
Il processo QAS prevede due fasi principali:

- **Addestramento basato sulla quantizzazione:** In questa fase, la rete neurale viene addestrata tenendo conto della quantizzazione, simulandola per imitarne gli effetti durante i passaggi "forward" e "backward". Ciò consente al modello di imparare a compensare gli errori di quantizzazione e migliorarne le prestazioni su hardware a bassa precisione. Per i dettagli, fare riferimento alla sezione QAT in Ottimizzazioni del modello.
- **Quantizzazione e ridimensionamento:** Dopo l'addestramento, il modello viene quantizzato in un formato a bassa precisione e i fattori di scala vengono regolati per ridurre al minimo gli errori di quantizzazione. I fattori di scala vengono scelti in base alla distribuzione dei pesi e delle attivazioni nel modello e vengono regolati per garantire che i valori quantizzati siano compresi nell'intervallo del formato a bassa precisione.

QAS viene utilizzato per superare le difficoltà di ottimizzazione dei modelli su dispositivi minuscoli senza dover effettuare la messa a punto degli iperparametri; QAS ridimensiona automaticamente i gradienti tensoriali con varie precisioni di bit. Ciò stabilizza il processo di addestramento e corrisponde all'accuratezza della precisione in virgola mobile.

12.3.2.2 Aggiornamenti Sparsi

Sebbene QAS consenta l'ottimizzazione di un modello quantizzato, utilizza una grande quantità di memoria, il che non è realistico per l'addestramento sul dispositivo. Quindi, gli aggiornamenti "spare" vengono utilizzati per ridurre l'ingombro di memoria del calcolo "full backward". Invece di potare i pesi per l'inferenza, l'aggiornamento sparso pota il gradiente durante la "backward propagation" [propagazione all'indietro] per aggiornare il modello in modo sparso. In altre parole, l'aggiornamento sparso salta i gradienti del calcolo di layer e sottotensori meno importanti.

Tuttavia, determinare lo schema di un aggiornamento sparso ottimale dato un budget di memoria vincolante può essere difficile a causa dell'ampio spazio di ricerca. Ad esempio, il modello MCUNet ha 43 layer convoluzionali e uno spazio di ricerca di circa 1030. Una tecnica per affrontare questo problema è l'analisi del contributo. L'analisi del contributo misura il miglioramento dell'accuratezza dai bias (aggiornamento degli ultimi bias rispetto al solo aggiornamento del classificatore) e pesi (aggiornamento del peso di un layer extra rispetto al solo aggiornamento del bias). Cercando di massimizzare questi miglioramenti, l'analisi del contributo deriva automaticamente uno schema di aggiornamento sparso ottimale per abilitare l'addestramento sul dispositivo.

12.3.2.3 Training Layer-Wise

Altri metodi oltre alla quantizzazione possono aiutare a ottimizzare le routine. Uno di questi metodi è l'addestramento "layer-wise". Un consumatore significativo di memoria dell'addestramento DNN è la backpropagation end-to-end, che richiede che tutte le feature map intermedie siano archiviate in modo che il modello possa calcolare i gradienti. Un'alternativa a questo approccio che riduce l'impronta di memoria dell'addestramento DNN è l'addestramento sequenziale "layer-by-layer" (T. Chen et al. 2016). Invece dell'addestramento end-to-end, l'addestramento di un singolo layer alla volta aiuta a evitare di dover archiviare le feature map intermedie.

12.3.2.4 Trading Computation for Memory

La strategia "trading computation for memory" [scambio di elaborazione per memoria] comporta il rilascio di parte della memoria utilizzata per archiviare i risultati intermedi. Invece, questi risultati possono essere ricalcolati in base alle necessità. È stato dimostrato che la riduzione della memoria in cambio di più elaborazione riduce l'impronta di memoria dell'addestramento DNN per adattarsi a quasi tutti i budget, riducendo al minimo anche i costi di elaborazione (Gruslys et al. 2016).

12.3.3 Sviluppo di Nuove Rappresentazioni dei Dati

La dimensionalità e il volume dei dati di training possono avere un impatto significativo sull'adattamento sul dispositivo. Quindi, un'altra tecnica per adattare i modelli su dispositivi con risorse limitate è quella di rappresentare i set di dati in modo più efficiente.

12.3.3.1 Compressione dei Dati

L'obiettivo della compressione dei dati è raggiungere elevate precisioni limitando al contempo la quantità di dati di training. Un metodo per raggiungere questo obiettivo è dare priorità alla complessità del campione: la quantità di dati di training necessari affinché l'algoritmo raggiunga una precisione target (Dhar et al. 2021).

Altri metodi più comuni di compressione dei dati si concentrano sulla riduzione della dimensionalità e del volume dei dati di training. Ad esempio, un approccio potrebbe sfruttare la sparsità della matrice per ridurre l'ingombro di memoria per l'archiviazione dei dati di training. I dati di training possono essere trasformati in un embedding a dimensione inferiore e fattorizzati in una matrice di dizionario moltiplicata per una matrice di coefficienti blocchi sparsi (Darvish Rouhani, Mirhoseini, e Koushanfar 2017). Un altro esempio potrebbe

riguardare la rappresentazione di parole provenienti da un ampio set di dati di training linguistica in un formato vettoriale più compresso (X. Li et al. 2016).

12.4 Il Transfer Learning

Il transfer learning è una tecnica in cui un modello sviluppato per un'attività specifica viene riutilizzato come punto di partenza per un modello su una seconda attività. Il transfer learning ci consente di sfruttare modelli pre-addestrati che hanno già appreso rappresentazioni utili da grandi set di dati e di perfezionarli per attività specifiche utilizzando set di dati più piccoli direttamente sul dispositivo. Ciò può ridurre significativamente le risorse di calcolo e il tempo necessari per l'addestramento dei modelli da zero.
Può essere compreso attraverso esempi intuitivi del mondo reale, come illustrato in Figura 12.4. La figura mostra scenari in cui le competenze di un dominio possono essere applicate per accelerare l'apprendimento in un campo correlato. Un esempio lampante è la relazione tra andare in bicicletta e andare in moto. Se si sa andare in bicicletta, si ha già l'abilità di stare in equilibrio su un veicolo a due ruote. La conoscenza di base di questa abilità rende significativamente più facile per te imparare a guidare una moto rispetto a qualcuno senza alcuna esperienza ciclistica. La figura illustra questo e altri scenari simili, dimostrando come l'apprendimento per trasferimento sfrutti le conoscenze esistenti per accelerare l'acquisizione di nuove competenze correlate.

Figura 12.4: Trasferimento di conoscenze tra attività. Fonte: Zhuang et al. (2021).

Prendiamo l'esempio di un'applicazione di sensore intelligente che utilizza l'intelligenza artificiale on-device per riconoscere gli oggetti nelle immagini acquisite dal dispositivo. Tradizionalmente, ciò richiederebbe l'invio dei dati dell'immagine a un server, dove un ampio modello di rete neurale elabora i dati e invia i risultati. Con l'intelligenza artificiale on-device, il modello viene archiviato ed eseguito direttamente sul dispositivo, eliminando la necessità di inviare dati a un server.
Per personalizzare il modello per le caratteristiche on-device, addestrare un modello di rete neurale da zero sul dispositivo sarebbe poco pratico a causa delle risorse di calcolo limitate e della durata della batteria. È qui che entra in gioco il "transfer learning" [apprendimento tramite trasferimento]. Invece di addestrare un modello da zero, possiamo prendere un modello pre-addestrato, come una rete neurale convoluzionale (CNN) o una rete di trasformatori addestrata su un ampio set di dati di immagini, e perfezionarlo per la nostra specifica attività di riconoscimento degli oggetti. Questa messa a punto può essere eseguita direttamente sul dispositivo utilizzando un set di dati più piccolo di immagini pertinenti all'attività. Sfruttando il modello pre-addestrato, possiamo ridurre le risorse di calcolo e il tempo necessari per il training, ottenendo comunque un'elevata precisione per l'attività di riconoscimento degli oggetti. Figura 12.5 illustra ulteriormente i vantaggi dell'apprendimento tramite trasferimento rispetto al training da zero.

Figura 12.5: Training da zero vs. apprendimento tramite trasferimento.

Il transfer learning è importante per rendere praticabile l'intelligenza artificiale on-device, consentendoci di sfruttare modelli pre-addestrati e di perfezionarli per attività specifiche, riducendo così le risorse di calcolo e il tempo necessari per il training. La combinazione di intelligenza artificiale sul dispositivo e il "transfer learning" apre nuove possibilità per applicazioni di intelligenza artificiale più attente alla privacy e più reattive alle esigenze degli utenti.

Il transfer learning ha rivoluzionato il modo in cui i modelli vengono sviluppati e distribuiti, sia nel cloud che nell'edge. Il transfer learning viene utilizzato nel mondo reale. Un esempio del genere è l'uso del transfer learning per sviluppare modelli di intelligenza artificiale in grado di rilevare e diagnosticare malattie da immagini mediche, come raggi X, scansioni MRI [risonanza magnetica] e TAC. Ad esempio, i ricercatori della Stanford University hanno sviluppato un modello di apprendimento di trasferimento in grado di rilevare il cancro nelle immagini della pelle con una precisione del 97% (Esteva et al. 2017). Questo modello è stato pre-addestrato su 1.28 milioni di immagini per classificare un'ampia gamma di oggetti e poi specializzato per il rilevamento del cancro tramite l'addestramento su un set di dati di immagini della pelle curato da dermatologi.

L'implementazione negli scenari di produzione può essere ampiamente categorizzata in due fasi: pre-distribuzione e post-distribuzione.

12.4.1 Specializzazione Pre-Distribuzione

Nella fase di pre-implementazione, il transfer learning funge da catalizzatore per accelerare il processo di sviluppo. Ecco come funziona tipicamente: Si immagini di creare un sistema per riconoscere diverse razze di cani. Invece di partire da zero, possiamo utilizzare un modello pre-addestrato che ha già padroneggiato il compito più ampio di riconoscere gli animali nelle immagini.

Questo modello pre-addestrato funge da solida base e contiene una vasta conoscenza acquisita da dati estesi. Quindi perfezioniamo questo modello utilizzando un set di dati specializzato contenente immagini di varie razze di cani. Questo processo di messa a punto adatta il modello alle nostre esigenze specifiche, ovvero identificare con precisione le razze di cani. Una volta perfezionato e convalidato per soddisfare i criteri di prestazione, questo modello specializzato è pronto per l'implementazione.

Ecco come funziona in pratica:

- **Inizia con un Modello Pre-Addestrato:** Si inizia selezionando un modello che è già stato addestrato su un set di dati completo, solitamente correlato a un'attività generale. Questo modello funge da base per l'attività in questione.

- **Finetuning:** Il modello pre-addestrato viene quindi perfezionato su un set di dati più piccolo e specifico per l'attività desiderata. Questo passaggio consente al modello di adattare e specializzare la sua conoscenza ai requisiti specifici dell'applicazione.
- **Validazione:** Dopo la messa a punto, il modello viene convalidato per garantire che soddisfi i criteri di prestazione per l'attività specializzata.
- **Deployment:** Una volta convalidato, il modello specializzato viene distribuito nell'ambiente di produzione.

Questo metodo riduce significativamente il tempo e le risorse di calcolo necessarie per addestrare un modello da zero (Pan e Yang 2010). Adottando l'apprendimento tramite trasferimento, i sistemi embedded possono raggiungere un'elevata precisione su attività specializzate senza la necessità di raccogliere dati estesi o di impiegare risorse di calcolo significative per l'addestramento da zero.

12.4.2 Adattamento Post-Distribuzione

L'implementazione su un dispositivo non deve necessariamente segnare il culmine del percorso educativo di un modello ML. Con l'avvento dell'apprendimento per trasferimento, apriamo le porte all'implementazione di modelli ML adattivi in scenari del mondo reale, soddisfacendo le esigenze personalizzate degli utenti.

Consideriamo un'applicazione reale in cui un genitore desidera identificare il proprio figlio in una raccolta di immagini di un evento scolastico sul proprio smartphone. In questo scenario, il genitore si trova di fronte alla sfida di localizzare il proprio figlio in mezzo alle immagini di molti altri bambini. L'apprendimento per trasferimento può essere impiegato qui per perfezionare il modello di un sistema embedded per questo compito unico e specializzato. Inizialmente, il sistema potrebbe utilizzare un modello generico addestrato per riconoscere i volti nelle immagini. Tuttavia, con l'apprendimento per trasferimento, il sistema può adattare questo modello per riconoscere le caratteristiche specifiche del figlio dell'utente.

Ecco come funziona:

1. **Raccolta Dati:** Il sistema embedded raccoglie immagini che includono il bambino, idealmente con l'input del genitore per garantire accuratezza e pertinenza. Ciò può essere fatto direttamente sul dispositivo, mantenendo la privacy dei dati dell'utente.
2. **Fine Tuning del Modello:** Il modello di riconoscimento facciale preesistente, che è stato addestrato su un set di dati ampio e diversificato, viene quindi perfezionato utilizzando le immagini del bambino appena raccolte. Questo processo adatta il modello per riconoscere le caratteristiche facciali specifiche del bambino, distinguendolo dagli altri bambini nelle immagini.
3. **Validazione:** Il modello rifinito viene poi convalidato per garantire che riconosca accuratamente il bambino in varie immagini. Ciò può comportare che il genitore verifichi le prestazioni del modello e fornisca feedback per ulteriori miglioramenti.
4. **Deployment:** Una volta convalidato, il modello adattato viene distribuito sul dispositivo, consentendo al genitore di identificare facilmente il proprio figlio nelle immagini senza doverle esaminare manualmente.

Questa personalizzazione al volo migliora l'efficacia del modello per il singolo utente, assicurando che tragga vantaggio dalla personalizzazione ML. Questo è, in parte, il modo in cui iPhotos o Google Photos funzionano quando ci chiedono di riconoscere un volto e poi, in base a queste informazioni, indicizzano tutte le foto di quel volto. Poiché l'apprendimento e l'adattamento avvengono sul dispositivo stesso, non ci sono rischi per la privacy personale. Le immagini dei genitori non vengono caricate su un server cloud o condivise con terze parti, proteggendo la privacy della famiglia e continuando a raccogliere i benefici di un modello ML

personalizzato. Questo approccio rappresenta un significativo passo avanti nella ricerca per fornire agli utenti soluzioni ML personalizzate che rispettino e sostengano la loro privacy.

12.4.3 Vantaggi

Il transfer learning è diventato una tecnica importante in ML e intelligenza artificiale, ed è particolarmente prezioso per diversi motivi.

1. **Scarsità di Dati:** In molti scenari reali, acquisire un set di dati etichettato sufficientemente grande per addestrare un modello ML da zero è complicato. Il transfer learning mitiga questo problema consentendo l'uso di modelli pre-addestrati che hanno già appreso funzionalità preziose da un vasto set di dati.
2. **Spese Computazionali:** Addestrare un modello da zero richiede risorse computazionali e tempo significativi, specialmente per modelli complessi come reti neurali profonde. Utilizzando il transfer learning, possiamo sfruttare il calcolo che è già stato eseguito durante l'addestramento del modello sorgente, risparmiando così tempo e potenza computazionale.
3. **Dati Annotati Limitati:** Per alcune attività specifiche, potrebbero essere disponibili ampi dati grezzi, ma il processo di etichettatura di tali dati per l'apprendimento supervisionato può essere costoso e richiedere molto tempo. Il transfer learning ci consente di utilizzare modelli pre-addestrati su un'attività correlata con dati etichettati, quindi richiedendo meno dati annotati per la nuova attività.

Ci sono vantaggi nel riutilizzare le funzionalità:

1. **Hierarchical Feature Learning:** I modelli di deep learning, in particolare le reti neurali convoluzionali (CNN), possono apprendere funzionalità gerarchiche. I layer inferiori in genere apprendono funzionalità generiche come bordi e forme, mentre quelli superiori apprendono funzionalità più complesse e specifiche per l'attività. Il transfer learning ci consente di riutilizzare le funzionalità generiche apprese da un modello e di perfezionare i livelli superiori per la nostra attività specifica.
2. **Aumento delle Prestazioni:** È stato dimostrato che il transfer learning aumenta le prestazioni dei modelli su attività con dati limitati. La conoscenza acquisita dall'attività dall'attività sorgente può fornire un prezioso punto di partenza e portare a una convergenza più rapida e a una maggiore accuratezza nell'attività target.

Esercizio 12.1: Il Transfer Learning

Si immagini di addestrare un'IA a riconoscere i fiori come un professionista, ma senza aver bisogno di un milione di immagini di fiori! Questo è il potere del transfer learning. In questo Colab, prenderemo un'IA che conosce già le immagini e le insegneremo a diventare un'esperta di fiori con meno sforzo. Prepararsi a rendere la propria IA più intelligente, non è più difficile!

Open in Colab

12.4.4 Concetti Fondamentali

Comprendere i concetti fondamentali del transfer learning è essenziale per utilizzare efficacemente questo potente approccio in ML. Qui, analizzeremo alcuni dei principi e dei componenti principali che stanno alla base del processo di transfer learning.

12.4.4.1 Attività di Origine e di Destinazione

Nel transfer learning, sono coinvolte due attività principali: l'attività di origine e quella di destinazione. L'attività di origine è quella per la quale il modello è già stato addestrato e ha appreso informazioni preziose. L'attività di destinazione è la nuova attività che vogliamo che il

modello esegua. L'obiettivo del transfer learning è sfruttare le conoscenze acquisite dall'attività di origine per migliorare le prestazioni nell'attività di destinazione. Supponiamo di avere un modello addestrato per riconoscere vari frutti nelle immagini (attività di origine) e di voler creare un nuovo modello per riconoscere diverse verdure nelle immagini (attività di destinazione). In tal caso, possiamo utilizzare il transfer learning per sfruttare le conoscenze acquisite durante l'attività di riconoscimento della frutta per migliorare le prestazioni del modello di riconoscimento della verdura.

12.4.4.2 Trasferimento della Rappresentazione

Il trasferimento della rappresentazione riguarda le rappresentazioni apprese (caratteristiche) dall'attività di origine all'attività di destinazione. Esistono tre tipi principali di trasferimento della rappresentazione:

- **Trasferimento di Istanza:** Implica il riutilizzo delle istanze di dati dall'attività di origine nell'attività di destinazione.
- **Trasferimento della Rappresentazione delle Feature:** Implica il trasferimento delle rappresentazioni di Feature [funzionalità] apprese dall'attività di origine all'attività di destinazione.
- **Trasferimento di Parametri:** Implica il trasferimento dei parametri appresi del modello (pesi) dall'attività di origine all'attività di destinazione.

Nell'elaborazione del linguaggio naturale, un modello addestrato per comprendere la sintassi e la grammatica di una lingua (attività di origine) può trasferire le sue rappresentazioni apprese a un nuovo modello progettato per eseguire l'analisi del sentiment (attività di destinazione).

12.4.4.3 Finetuning

Il "finetuning" [messa a punto] è il processo di regolazione dei parametri di un modello pre-addestrato per adattarlo all'attività di destinazione. In genere, ciò comporta l'aggiornamento dei pesi dei layer del modello, in particolare degli ultimi layer, per rendere il modello più pertinente per la nuova attività. Nella classificazione delle immagini, un modello pre-addestrato su un set di dati generale come ImageNet (attività di origine) può essere messo a punto regolando i pesi dei suoi livelli per ottenere buone prestazioni in un'attività di classificazione specifica, come il riconoscimento di specie animali specifiche (attività di destinazione).

12.4.4.4 Estrazione delle Feature

L'estrazione delle "feature" caratteristiche comporta l'utilizzo di un modello pre-addestrato come estrattore di feature fisse, in cui l'output dei layer intermedi del modello viene utilizzato come feature per l'attività di destinazione. Questo approccio è particolarmente utile quando l'attività di destinazione ha un set di dati di piccole dimensioni, poiché le feature apprese dal modello pre-addestrato possono migliorare significativamente le prestazioni. Nell'analisi delle immagini mediche, un modello pre-addestrato su un ampio set di dati di immagini mediche generali (attività di origine) può essere utilizzato come estrattore di feature per fornire funzionalità preziose per un nuovo modello progettato per riconoscere specifici tipi di tumori nelle immagini radiografiche (attività di destinazione).

12.4.5 Tipi di Apprendimento Tramite Trasferimento

L'apprendimento tramite trasferimento può essere classificato in tre tipi principali in base alla natura delle attività e dei dati di origine e di destinazione. Esploriamo ciascun tipo in dettaglio:

12.4.5.1 Apprendimento Tramite Trasferimento Induttivo

Nell'apprendimento tramite trasferimento induttivo, l'obiettivo è apprendere la funzione predittiva di destinazione con l'aiuto dei dati di origine. In genere comporta la messa a punto di un modello pre-addestrato sull'attività di destinazione con dati etichettati disponibili. Un esempio comune di apprendimento tramite trasferimento induttivo sono le attività di classificazione delle immagini. Ad esempio, un modello pre-addestrato sul set di dati ImageNet (attività di origine) può essere messo a punto per classificare tipi specifici di uccelli (attività di destinazione) utilizzando un set di dati etichettato più piccolo di immagini di uccelli.

12.4.5.2 Apprendimento Tramite Trasferimento Transduttivo

L'apprendimento tramite trasferimento transduttivo comporta l'utilizzo di dati di origine e destinazione, ma solo dell'attività di origine. L'obiettivo principale è trasferire la conoscenza dal dominio di origine al dominio di destinazione, anche se le attività rimangono le stesse. L'analisi del "sentiment" per diverse lingue può servire come esempio di apprendimento tramite trasferimento transduttivo. Un modello addestrato per eseguire l'analisi del sentiment in inglese (attività di origine) può essere adattato per eseguire l'analisi del sentiment in un'altra lingua, come il francese (attività di destinazione), sfruttando set di dati paralleli di frasi in inglese e francese con gli stessi sentimenti.

12.4.5.3 Apprendimento con Trasferimento Non Supervisionato

L'apprendimento con trasferimento non supervisionato viene utilizzato quando le attività di origine e di destinazione sono correlate, ma non sono disponibili dati etichettati per l'attività di destinazione. L'obiettivo è sfruttare la conoscenza acquisita dall'attività di origine per migliorare le prestazioni nell'attività di destinazione, anche senza dati etichettati. Un esempio di apprendimento di trasferimento non supervisionato è la modellazione degli argomenti nei dati di testo. Un modello addestrato per estrarre argomenti da articoli di notizie (attività di origine) può essere adattato per estrarre argomenti da post sui social media (attività di destinazione) senza aver bisogno di dati etichettati per i post sui social media.

12.4.5.4 Confronto e Compromessi

Sfruttando questi diversi tipi di apprendimento per trasferimento, i professionisti possono scegliere l'approccio che meglio si adatta alla natura dei loro compiti e ai dati disponibili, portando infine a modelli di ML più efficaci ed efficienti. Quindi, in sintesi:

- **Induttivo:** Diversi compiti di origine e destinazione, domini diversi
- **Trasduttivo:** diversi compiti di origine e destinazione, stesso dominio
- **Non supervisionato:** dati di origine non etichettati, trasferisce le rappresentazioni delle feature

Tabella 12.1 presenta una matrice che delinea in modo un po' più dettagliato le somiglianze e le differenze tra i tipi di apprendimento per trasferimento:

Tabella 12.1: Confronto dei tipi di apprendimento per trasferimento.

Aspetto	Apprendimento Induttivo per Trasferimento	Apprendimento Trasduttivo per Trasferimento	Apprendimento Non Supervisionato
Dati Etichettati per l'Attività di Destinazione	Obbligatorio	Non obbligatorio	Non obbligatorio
Attività di origine	Può essere diversa	Lo stesso	Lo stesso o diverso
Attività di destinazione	Può essere diversa	Lo stesso	Può essere diverso

Obiettivo	Migliorare le prestazioni dell'attività target con i dati sorgente	Trasferisci la conoscenza dal dominio sorgente a quello target	Sfrutta l'attività sorgente per migliorare le prestazioni dell'attività target senza dati etichettati
Esempio	Da ImageNet alla classificazione degli uccelli	Analisi del sentiment in diverse lingue	Modellazione degli argomenti per diversi dati di testo

12.4.6 Vincoli e Considerazioni

Quando si intraprende un apprendimento per trasferimento, ci sono diversi fattori che devono essere considerati per garantire un trasferimento di conoscenze di successo e prestazioni del modello. Ecco una ripartizione di alcuni fattori chiave:

12.4.6.1 Somiglianza dei Domini

La similarità di dominio si riferisce a quanto strettamente correlati sono i domini di origine e di destinazione. Più simili sono i domini, più è probabile che l'apprendimento per trasferimento abbia successo. Trasferire la conoscenza da un modello addestrato su immagini di scene esterne (dominio di origine) a un nuovo compito che prevede il riconoscimento di oggetti in scene interne (dominio di destinazione) potrebbe avere più successo rispetto al trasferire la conoscenza da scene esterne a un compito che prevede l'analisi del testo, poiché i domini (immagini vs. testo) sono piuttosto diversi.

12.4.6.2 Similarità dell'Attività

La similarità dell'attività si riferisce a quanto strettamente correlati sono le attività di origine e di destinazione. È probabile che attività simili traggano maggiori benefici dall'apprendimento per trasferimento. Un modello addestrato per riconoscere diverse razze di cani (attività di origine) può essere adattato più facilmente per riconoscere diverse razze di gatti (attività di destinazione) rispetto a quanto possa essere adattato per eseguire un'attività completamente diversa come la traduzione di una lingua.

12.4.6.3 Qualità e Quantità dei Dati

La qualità e la quantità dei dati disponibili per il compito di destinazione possono avere un impatto significativo sul successo dell'apprendimento per trasferimento. Più dati di alta qualità possono comportare migliori prestazioni del modello. Supponiamo di avere un ampio set di dati con immagini chiare e ben etichettate per riconoscere specie di uccelli specifiche. In tal caso, il processo di apprendimento per trasferimento avrà probabilmente più successo rispetto a un set di dati piccolo e rumoroso.

12.4.6.4 Sovrapposizione dello Spazio delle Feature

La sovrapposizione dello spazio delle feature si riferisce a quanto bene le feature apprese dal modello sorgente si allineano con quelle necessarie per l'attività di destinazione. Una maggiore sovrapposizione può portare a un apprendimento per trasferimento più efficace. Un modello addestrato su immagini ad alta risoluzione (attività di origine) potrebbe non trasferirsi bene a un'attività di destinazione che coinvolge immagini a bassa risoluzione, poiché lo spazio delle feature (alta risoluzione rispetto a bassa risoluzione) è diverso.

12.4.6.5 Complessità del Modello

Anche la complessità del modello sorgente può influire sul successo dell'apprendimento per trasferimento. A volte, un modello più semplice potrebbe trasferirsi meglio di uno complesso, poiché è meno probabile che si adatti eccessivamente all'attività di origine. Ad esempio, un semplice modello di rete neurale convoluzionale (CNN) addestrato su dati di immagini (attività di origine) può essere trasferito con maggiore successo a una nuova attività di classificazione di immagini (attività di destinazione) rispetto a una CNN complessa con molti

layer, poiché è meno probabile che il modello più semplice si adatti eccessivamente all'attività di origine.
Considerando questi fattori, i professionisti del ML possono prendere decisioni informate su quando e come utilizzare l'apprendimento per trasferimento, portando infine a prestazioni del modello più efficaci nell'attività di destinazione. Il successo dell'apprendimento per trasferimento dipende dal grado di similarità tra i domini di origine e di destinazione. L'overfitting [adattamento eccessivo] è rischioso, soprattutto quando la messa a punto avviene su un set di dati limitato. Sul fronte computazionale, alcuni modelli pre-addestrati, a causa delle loro dimensioni, potrebbero non adattarsi comodamente ai vincoli di memoria di alcuni dispositivi o potrebbero essere eseguiti in modo proibitivamente lento. Nel tempo, con l'evoluzione dei dati, c'è il potenziale per la "drift" [deriva] del modello, che indica la necessità di un riaddestramento periodico o di un adattamento continuo.
Scoprire di più sull'apprendimento per trasferimento in Video 12.1 di seguito.

Video 12.1: Il Transfer Learning

https://www.youtube.com/watch?v=FQM13HkEfBk

12.5 Apprendimento Automatico Federato

12.5.1 Panoramica dell'Apprendimento Federato

L'Internet moderna è piena di grandi reti di dispositivi connessi. Che si tratti di telefoni cellulari, termostati, smart speaker o altri prodotti IoT, innumerevoli dispositivi edge sono una miniera d'oro per dati iperpersonalizzati e ricchi. Tuttavia, con quei dati ricchi arriva una serie di problemi con il trasferimento delle informazioni e la privacy. Costruire un set di dati di training nel cloud da questi dispositivi comporterebbe un'ampia larghezza di banda, costi per il trasferimento dati e violazione della privacy degli utenti.
L'apprendimento federato offre una soluzione a questi problemi: addestrare i modelli parzialmente sui dispositivi edge e comunicare solo gli aggiornamenti al cloud. Nel 2016, un team di Google ha progettato un'architettura per l'apprendimento federato che tenta di risolvere questi problemi. Nel loro articolo iniziale, McMahan et al. (2017b) delinea un algoritmo di apprendimento federato di principio chiamato FederatedAveraging, mostrato in Figura 12.6. In particolare, FederatedAveraging esegue la "stochastic gradient descent (SGD) [discesa del gradiente stocastico] su diversi dispositivi edge. In questo processo, ogni dispositivo calcola un gradiente $g_k = \nabla F_k(w_t)$ che viene poi applicato per aggiornare i pesi lato server come (con η come tasso di apprendimento su k client):

$$w_{t+1} \rightarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$$

Questo riassume l'algoritmo di base per l'apprendimento federato sulla destra. Per ogni round di addestramento, il server prende un set casuale di dispositivi client e chiama ogni client per addestrare sul suo batch locale usando i pesi lato server più recenti. Tali pesi vengono poi restituiti al server, dove vengono raccolti individualmente e calcolati per aggiornare i pesi del modello globale.

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
  initialize w_0
  for each round $t = 1, 2, \ldots$ **do**
    $m \leftarrow \max(C \cdot K, 1)$
    $S_t \leftarrow$ (random set of m clients)
    for each client $k \in S_t$ **in parallel do**
      $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
    $m_t \leftarrow \sum_{k \in S_t} n_k$
    $w_{t+1} \leftarrow \sum_{k \in S_t} \frac{n_k}{m_t} w_{t+1}^k$ *// Erratum*[4]

ClientUpdate(k, w)**:** *// Run on client k*
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
  for each local epoch i from 1 to E **do**
    for batch $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$
  return w to server

Figura 12.6: Algoritmo FederatedAverage proposto da Google. Fonte: McMahan et al. (2017).

Con questa struttura proposta, ci sono alcuni vettori chiave per ottimizzare ulteriormente l'apprendimento federato. Descriveremo ciascuno di essi nelle seguenti sottosezioni. Video 12.2 fornisce una panoramica dell'apprendimento federato.

Video 12.2: Il Transfer Learning

https://www.youtube.com/watch?v=zqv1eELa7fs

Figura 12.7 delinea l'impatto trasformativo dell'apprendimento federato sull'apprendimento on-device.

Figura 12.7: L'apprendimento federato sta rivoluzionando l'apprendimento on-device.

12.5.2 Efficienza della Comunicazione

Uno dei principali colli di bottiglia nell'apprendimento federato è la comunicazione. Ogni volta che un client addestra il modello, deve comunicare i propri aggiornamenti al server. Analogamente, una volta che il server ha calcolato la media di tutti gli aggiornamenti, deve inviarli al client. Ciò comporta enormi costi di larghezza di banda e risorse su grandi reti di

milioni di dispositivi. Con l'avanzare del campo dell'apprendimento federato, sono state sviluppate alcune ottimizzazioni per ridurre al minimo questa comunicazione. Per affrontare l'ingombro del modello, i ricercatori hanno sviluppato tecniche di compressione del modello. Nel protocollo client-server, l'apprendimento federato può anche ridurre al minimo la comunicazione tramite la condivisione selettiva degli aggiornamenti sui client. Infine, anche tecniche di aggregazione efficienti possono semplificare il processo di comunicazione.

12.5.3 Compressione del Modello

Nell'apprendimento federato standard, il server comunica l'intero modello a ciascun client, quindi il client invia tutti i pesi aggiornati. Ciò significa che il modo più semplice per ridurre l'ingombro di memoria e comunicazione del client è ridurre al minimo le dimensioni del modello che deve essere comunicato. Possiamo impiegare tutte le strategie di ottimizzazione del modello discusse in precedenza per farlo.
Nel 2022, un altro team di Google ha proposto che ogni client comunichi tramite un formato compresso e decomprima il modello al volo per l'addestramento (Yang et al. 2023), allocando e deallocando l'intera memoria per il modello solo per un breve periodo durante l'addestramento. Il modello viene compresso tramite una gamma di diverse strategie di quantizzazione elaborate nel loro documento. Nel frattempo, il server può aggiornare il modello non compresso decomprimendolo e applicando gli aggiornamenti man mano che arrivano.

12.5.4 Condivisione Selettiva degli Aggiornamenti

Esistono molti metodi per condividere selettivamente gli aggiornamenti. Il principio generale è che la riduzione della porzione del modello che i client stanno addestrando lato edge riduce la memoria necessaria per l'addestramento e la dimensione della comunicazione con il server. Nell'apprendimento federato di base, il client addestra l'intero modello. Ciò significa che quando un client invia un aggiornamento al server, ha gradienti per ogni peso nella rete. Tuttavia, non possiamo semplicemente ridurre la comunicazione inviando parti di quei gradienti da ogni client al server perché i gradienti fanno parte di un intero aggiornamento necessario per migliorare il modello. Invece, si deve progettare architettonicamente il modello in modo che ogni client addestri solo una piccola parte del modello più ampio, riducendo la comunicazione totale e ottenendo comunque il vantaggio dell'addestramento sui dati del client. Shi e Radu (2022) applica questo concetto a una CNN suddividendo il modello globale in due parti: una superiore e una inferiore, come mostrato in Zhiyong Chen e Xu (2023).

Figura 12.8: Suddivisione dell'architettura del modello per la condivisione selettiva. Fonte: Shi et al., (2022).

La parte inferiore è progettata per concentrarsi sulle funzionalità generiche nel set di dati, mentre la parte superiore, addestrata su tali funzionalità generiche, è progettata per essere più sensibile alle mappe di attivazione. Ciò significa che la parte inferiore del modello viene addestrata tramite la media federata standard su tutti i client. Nel frattempo, la parte superiore del modello viene addestrata interamente sul lato server dalle mappe di attivazione generate dai client. Questo approccio riduce drasticamente la comunicazione per il modello, rendendo comunque la rete robusta a vari tipi di input trovati nei dati sui dispositivi client.

12.5.5 Aggregazione Ottimizzata

Oltre a ridurre il sovraccarico di comunicazione, l'ottimizzazione della funzione di aggregazione può migliorare la velocità e l'accuratezza dell'addestramento del modello in determinati casi d'uso di apprendimento federato. Mentre lo standard per l'aggregazione è solo la media, vari altri approcci possono migliorare l'efficienza, l'accuratezza e la sicurezza del modello.

Un'alternativa è la "media ritagliata", che limita gli aggiornamenti del modello entro un intervallo specifico. Un'altra strategia per preservare la sicurezza è l'aggregazione media della privacy differenziale. Questo approccio integra la privacy differenziale nella fase di aggregazione per proteggere le identità dei client. Ogni client aggiunge uno strato di rumore casuale ai propri aggiornamenti prima di comunicare al server. Il server si aggiorna quindi con gli aggiornamenti rumorosi, il che significa che la quantità di rumore deve essere regolata attentamente per bilanciare privacy e precisione.

Oltre ai metodi di aggregazione che migliorano la sicurezza, ci sono diverse modifiche ai metodi di aggregazione che possono migliorare la velocità di training e le prestazioni aggiungendo metadati client insieme agli aggiornamenti del peso. Il "Momentum aggregation" è una tecnica che aiuta ad affrontare il problema della convergenza. Nell'apprendimento federato, i dati client possono essere estremamente eterogenei a seconda dei diversi ambienti in cui vengono utilizzati i dispositivi. Ciò significa che molti modelli con dati eterogenei potrebbero aver bisogno di aiuto per convergere. Ogni client memorizza localmente un termine di "momentum", che traccia il ritmo del cambiamento su diversi aggiornamenti. Con i client che comunicano questo "momentum", il server può tenere conto della velocità di cambiamento di ogni aggiornamento quando si modifica il modello globale per accelerare la convergenza. Allo stesso modo, l'aggregazione ponderata può tenere conto delle prestazioni del client o di altri parametri come il tipo di dispositivo o la potenza della connessione di rete per regolare il peso con cui il server dovrebbe incorporare gli aggiornamenti del modello. Ulteriori descrizioni di algoritmi di aggregazione specifici sono fornite da Moshawrab et al. (2023).

12.5.6 Gestione dei Dati non-IID

Quando si utilizza l'apprendimento federato per addestrare un modello su molti dispositivi client, è conveniente considerare i dati come indipendenti e distribuiti in modo identico (IID) su tutti i client. Quando i dati sono IID, il modello convergerà più velocemente e funzionerà meglio perché ogni aggiornamento locale su un dato client è più rappresentativo del set di dati più ampio. Questo semplifica l'aggregazione, poiché è possibile calcolare direttamente la media di tutti i client. Tuttavia, differisce dal modo in cui i dati spesso appaiono nel mondo reale. Si considerino alcuni dei seguenti modi in cui i dati possono essere non IID:

- Imparando su un set di dispositivi di monitoraggio sanitari, diversi modelli di dispositivi potrebbero significare diverse qualità e proprietà dei sensori. Ciò significa che sensori e dispositivi di bassa qualità possono produrre dati e, pertanto, aggiornamenti del modello nettamente diversi da quelli di alta qualità
- Una tastiera intelligente addestrata per eseguire la correzione automatica. Se si ha una quantità sproporzionata di dispositivi da una determinata regione, lo slang, la struttura della frase o persino il linguaggio che stavano usando potrebbero deviare più aggiornamenti verso un certo stile di digitazione
- Se si hanno sensori per la fauna selvatica in aree remote, la connettività potrebbe non essere distribuita equamente, facendo sì che alcuni client in determinate regioni non siano in grado di inviare più aggiornamenti rispetto ad altri. Se quelle regioni hanno un'attività di fauna selvatica diversa da alcune specie, ciò potrebbe distorcere gli aggiornamenti verso quegli animali

Esistono alcuni approcci per affrontare i dati non IID nell'apprendimento federato. Uno potrebbe essere quello di modificare l'algoritmo di aggregazione. Se si utilizza un algoritmo di aggregazione ponderato, è possibile regolarlo in base a diverse proprietà del client come regione, proprietà del sensore o connettività (Y. Zhao et al. 2018).

12.5.7 Selezione del Client

Considerando tutti i fattori che influenzano l'efficacia dell'apprendimento federato, come i dati IID e la comunicazione, la selezione del client è una componente chiave per garantire che un sistema si alleni bene. La selezione dei client sbagliati può distorcere il dataset, con conseguenti dati non IID. Analogamente, la scelta casuale di client con cattive connessioni di rete può rallentare la comunicazione. Pertanto, è necessario considerare diverse caratteristiche chiave quando si seleziona il sottoinsieme corretto di client.
Quando si selezionano i client, ci sono tre componenti principali da considerare: eterogeneità dei dati, allocazione delle risorse e costo della comunicazione. Possiamo selezionare i client in base alle metriche proposte in precedenza nella sezione non IID per affrontare l'eterogeneità dei dati. Nell'apprendimento federato, tutti i dispositivi possono avere diverse quantità di elaborazione, con il risultato che alcuni sono più inefficienti di altri nell'addestramento. Quando si seleziona un sottoinsieme di client per l'addestramento, si deve considerare un equilibrio tra eterogeneità dei dati e risorse disponibili. In uno scenario ideale, è sempre possibile selezionare il sottoinsieme di client con le maggiori risorse. Tuttavia, questo potrebbe distorcere il set di dati, quindi è necessario trovare un equilibrio. Le differenze di comunicazione aggiungono un altro layer; si desidera evitare di essere bloccati dall'attesa che i dispositivi con connessioni scadenti trasmettano tutti i loro aggiornamenti. Pertanto, è anche necessario considerare la scelta di un sottoinsieme di dispositivi diversi ma ben collegati.

12.5.8 L'Esempio di Gboard

Un esempio primario di un sistema di apprendimento federato distribuito è la tastiera di Google, Gboard, per dispositivi Android. Nell'implementare l'apprendimento federato per la tastiera, Google si è concentrata sull'impiego di tecniche di privacy differenziali per proteggere i dati e l'identità dell'utente. Gboard sfrutta modelli linguistici per diverse funzionalità chiave, come Next Word Prediction (NWP), Smart Compose (SC) e On-The-Fly rescoring (OTF) (Z. Xu et al. 2023), come mostrato in Figura 12.9.
NWP anticiperà la parola successiva che l'utente tenta di digitare in base a quella precedente. SC fornisce suggerimenti in linea per velocizzare la digitazione in base a ciascun carattere. OTF riclassificherà le parole successive proposte in base al processo di digitazione attivo. Tutti e tre questi modelli devono essere eseguiti rapidamente sull'edge e l'apprendimento federato può accelerare l'addestramento sui dati degli utenti. Tuttavia, caricare ogni parola digitata da un utente sul cloud per l'addestramento costituirebbe una violazione massiccia della privacy. Pertanto, l'apprendimento federato enfatizza la privacy differenziale, che protegge l'utente consentendo al contempo una migliore esperienza utente.

Figura 12.9: Funzionalità di Google G Board. Fonte: Zheng et al., (2023).

Per raggiungere questo obiettivo, Google ha impiegato il suo algoritmo DP-FTRL, che fornisce una garanzia formale che i modelli addestrati non memorizzeranno dati o identità utente specifici. La progettazione del sistema dell'algoritmo è mostrata in Figura 12.10. DP-FTRL, combinato con l'aggregazione sicura, crittografa gli aggiornamenti del modello e fornisce un equilibrio ottimale tra privacy e utilità. Inoltre, il clipping adattivo viene applicato nel processo di aggregazione per limitare l'impatto dei singoli utenti sul modello globale (passaggio 3 in Figura 12.10). Combinando tutte queste tecniche, Google può perfezionare continuamente la sua tastiera preservando al contempo la privacy dell'utente in un modo formalmente dimostrabile.

Figura 12.10: Privacy Differenziale in G Board. Fonte: Zheng et al., (2023).

Esercizio 12.2: Apprendimento Federato - Generazione di Testo

Avete mai usato quelle tastiere intelligenti che suggeriscono la parola successiva? Con l'apprendimento federato, possiamo renderle ancora migliori senza sacrificare la privacy. In questo Colab, insegneremo a un'IA a prevedere le parole tramite l'addestramento su dati di testo distribuiti su più dispositivi. Prepariamoci a rendere la digitazione ancora più fluida!

Open in Colab

Esercizio 12.3: Apprendimento Federato - Classificazione delle Immagini

Vogliamo addestrare un'IA esperta di immagini senza inviare le proprie foto al cloud? L'apprendimento federato è la risposta! In questo Colab, addestreremo un modello su più dispositivi, ognuno dei quali apprende dalle proprie immagini. La privacy è protetta e il lavoro di squadra fa funzionare il sogno dell'IA!

Open in Colab

12.5.9 Benchmarking Federated Learning: MedPerf

I dispositivi medici rappresentano uno degli esempi più ricchi di dati edge. Questi dispositivi memorizzano alcuni dei dati utente più personali, offrendo allo stesso tempo progressi significativi nel trattamento personalizzato e una maggiore accuratezza nell'IA medica. Questa combinazione di dati sensibili e potenziale di innovazione rende i dispositivi medici un caso d'uso ideale per l'apprendimento federato.

Uno sviluppo chiave in questo campo è MedPerf, una piattaforma open source progettata per il benchmarking dei modelli utilizzando la valutazione federata (Karargyris et al. 2023). MedPerf va oltre il tradizionale apprendimento federato portando il modello sui dispositivi periferici per testare i dati personalizzati mantenendo la privacy. Questo approccio consente a un comitato di benchmark di valutare vari modelli in scenari reali su dispositivi edge senza compromettere l'anonimato del paziente.

La piattaforma MedPerf, descritta in dettaglio in uno studio recente (https://doi.org/10.1038/s42256-023-00652-2), dimostra come le tecniche federate possano essere applicate non solo all'addestramento dei modelli, ma anche alla valutazione e al benchmarking dei modelli. Questo progresso è particolarmente cruciale nel campo medico, dove l'equilibrio tra lo sfruttamento di grandi set di dati per migliorare le prestazioni dell'IA e la protezione della privacy individuale è di fondamentale importanza.

12.6 Problemi di Sicurezza

L'esecuzione di training e adattamento del modello ML sui dispositivi degli utenti finali introduce anche rischi per la sicurezza che devono essere affrontati. Alcune preoccupazioni chiave per la sicurezza includono:

- **Esposizione di dati privati:** I dati di training potrebbero essere trapelati o rubati dai dispositivi
- **Avvelenamento dei dati:** Gli avversari possono manipolare i dati di training per degradare le prestazioni del modello
- **Estrazione del modello:** Degli aggressori potrebbero tentare di rubare i parametri del modello addestrato
- **Inferenza di appartenenza:** I modelli potrebbero rivelare la partecipazione di dati di utenti specifici

- **Attacchi di evasione:** Input appositamente creati possono causare una classificazione errata

Qualsiasi sistema che esegue l'apprendimento sul dispositivo introduce preoccupazioni per la sicurezza, poiché potrebbe esporre vulnerabilità in modelli su larga scala. Numerosi rischi per la sicurezza sono associati a qualsiasi modello ML, ma questi rischi hanno conseguenze specifiche per l'apprendimento on-device. Fortunatamente, esistono metodi per mitigare questi rischi e migliorare le prestazioni reali dell'apprendimento su dispositivo.

12.6.1 Avvelenamento dei Dati

L'apprendimento automatico on-device introduce sfide uniche per la sicurezza dei dati rispetto all'addestramento tradizionale basato su cloud. In particolare, gli attacchi di avvelenamento dei dati rappresentano una seria minaccia durante l'apprendimento su dispositivo. Gli avversari possono manipolare i dati di training per degradare le prestazioni del modello quando vengono distribuiti.
Esistono diverse tecniche di attacco di avvelenamento dei dati:

- **Label Flipping:** Comporta l'applicazione di etichette errate ai campioni. Ad esempio, nella classificazione delle immagini, le foto di gatti possono essere etichettate come cani per confondere il modello. Anche capovolgere il 10% delle etichette può avere conseguenze significative sul modello.
- **Inserimento dei Dati:** Introduce input falsi o distorti nel set di training. Ciò potrebbe includere immagini pixelate, audio rumoroso o testo distorto.
- **Corruzione Logica:** Altera i [pattern] sottostanti (https://www.worldscientific.com/doi/10.1142/S0218001414600027) nei dati per fuorviare il modello. Nell'analisi del "sentiment", le recensioni altamente negative possono essere contrassegnate come positive tramite questa tecnica. Per questo motivo, recenti sondaggi hanno dimostrato che molte aziende hanno più paura dell'avvelenamento dei dati rispetto ad altre preoccupazioni di ML avversarie.

Ciò che rende l'avvelenamento dei dati allarmante è il modo in cui sfrutta la discrepanza tra set di dati curati e dati di training in tempo reale. Consideriamo un set di dati di foto di gatti raccolti da Internet. Nelle settimane successive, quando questi dati addestrano un modello on-device, le nuove foto di gatti sul Web differiscono in modo significativo.
Con l'avvelenamento dei dati, gli aggressori acquistano domini e caricano contenuti che influenzano una parte dei dati di training. Anche piccole modifiche ai dati hanno un impatto significativo sul comportamento appreso dal modello. Di conseguenza, l'avvelenamento può instillare pregiudizi razzisti, sessisti o altri pregiudizi dannosi se non controllato.
Microsoft Tay è un chatbot lanciato da Microsoft nel 2016. È stato progettato per imparare dalle sue interazioni con gli utenti su piattaforme di social media come Twitter. Sfortunatamente, Microsoft Tay è diventato un esempio lampante di avvelenamento dei dati nei modelli di ML. Entro 24 ore dal suo lancio, Microsoft ha dovuto mettere Tay offline perché aveva iniziato a produrre messaggi offensivi e inappropriati, tra cui incitamento all'odio e commenti razzisti. Ciò è accaduto perché alcuni utenti sui social media hanno intenzionalmente fornito a Tay input dannosi e offensivi, da cui il chatbot ha poi imparato e incorporato nelle sue risposte.
Questo incidente è un chiaro esempio di avvelenamento dei dati, poiché i malintenzionati hanno intenzionalmente manipolato i dati utilizzati per addestrare il chatbot e modellarne le risposte. L'avvelenamento dei dati ha portato il chatbot ad adottare pregiudizi dannosi e a produrre output che i suoi sviluppatori non avevano previsto. Dimostra come anche piccole quantità di dati creati in modo dannoso possano avere un impatto significativo sul

comportamento dei modelli ML e sottolinea l'importanza di implementare solidi meccanismi di filtraggio e convalida dei dati per impedire che tali incidenti si verifichino.
Tali pregiudizi potrebbero avere pericolosi impatti nel mondo reale. La convalida rigorosa dei dati, il rilevamento delle anomalie e il monitoraggio della provenienza dei dati sono misure difensive fondamentali. L'adozione di framework come Five Safes garantisce che i modelli siano addestrati su dati rappresentativi di alta qualità (Desai et al. 2016).
L'avvelenamento dei dati è una preoccupazione urgente per l'apprendimento sicuro sul dispositivo poiché i dati dell'endpoint non possono essere facilmente monitorati in tempo reale. Se ai modelli viene consentito di adattarsi da soli, corriamo il rischio che il dispositivo agisca in modo dannoso. Tuttavia, è necessaria una ricerca continua sull'apprendimento automatico avversario per sviluppare soluzioni efficaci per rilevare e mitigare tali attacchi ai dati.

12.6.2 Attacchi Avversari

Durante la fase di addestramento, gli aggressori potrebbero iniettare dati dannosi nel dataset di training, il che può alterare sottilmente il comportamento del modello. Ad esempio, un aggressore potrebbe aggiungere immagini di gatti etichettati come cani a un set di dati utilizzato per addestrare un modello di classificazione delle immagini. Se fatto in modo intelligente, l'accuratezza del modello potrebbe non diminuire in modo significativo e l'attacco potrebbe essere notato. Il modello classificherebbe quindi erroneamente alcuni gatti come cani, il che potrebbe avere conseguenze a seconda dell'applicazione.
In un sistema di telecamere di sicurezza embedded, ad esempio, ciò potrebbe consentire a un intruso di evitare il rilevamento indossando uno specifico pattern che il modello è stato ingannato a classificare come non minaccioso.
Durante la fase di inferenza, gli aggressori possono utilizzare esempi avversari per ingannare il modello. Gli esempi avversari sono input che sono stati leggermente alterati in un modo da far sì che il modello faccia previsioni errate. Ad esempio, un aggressore potrebbe aggiungere una piccola quantità di rumore a un'immagine in un modo che un sistema di riconoscimento facciale identifichi erroneamente una persona. Questi attacchi possono essere particolarmente preoccupanti nelle applicazioni in cui è in gioco la sicurezza, come i veicoli autonomi. Un esempio concreto di ciò è quando i ricercatori sono riusciti a far sì che un sistema di riconoscimento della segnaletica stradale classificasse erroneamente un segnale di stop come un segnale di limite di velocità. Questo tipo di classificazione errata potrebbe causare incidenti se si verificasse in un sistema di guida autonoma nel mondo reale.
Per mitigare questi rischi, possono essere impiegate diverse difese:

- **Validazione e Sanificazione dei Dati:** Prima di incorporare nuovi dati nel dataset di addestramento, questi devono essere convalidati e sanificati a fondo per garantire che non siano dannosi.
- **Addestramento Avversario:** Il modello può essere addestrato su esempi avversari per renderlo più robusto a questi tipi di attacchi.
- **Validazione degli Input:** Durante l'inferenza, gli input devono essere convalidati per garantire che non siano stati manipolati per creare esempi avversari.
- **Audit e Monitoraggio Regolari:** L'audit e il monitoraggio regolari del comportamento del modello possono aiutare a rilevare e mitigare gli attacchi avversari. Tuttavia, è più facile a dirsi che a farsi nel contesto di piccoli sistemi ML. Spesso è difficile monitorare i sistemi ML embedded all'endpoint a causa delle limitazioni della larghezza di banda della comunicazione, di cui parleremo nel capitolo MLOps.

Comprendendo i potenziali rischi e implementando queste difese, possiamo contribuire a proteggere il training on-device all'endpoint/edge e mitigare l'impatto degli attacchi avversari.

La maggior parte delle persone confonde facilmente l'avvelenamento dei dati e gli attacchi avversari. Quindi Tabella 12.2 confronta l'avvelenamento dei dati e gli attacchi avversari:

Tabella 12.2: Confronto tra avvelenamento dei dati e attacchi avversari.

Aspetto	Avvelenamento dei dati	Attacchi avversari
Tempistica	Fase di addestramento	Fase di inferenza
Target	Dati di addestramento	Dati di input
Obiettivo	Influenza negativamente le prestazioni del modello	Causa previsioni errate
Metodo	Inserire esempi dannosi nei dati di training, spesso con etichette errate	Aggiungere rumore attentamente elaborato ai dati di input
Esempio	Aggiungere immagini di gatti etichettati come cani a un set di dati utilizzato per addestrare un modello di classificazione delle immagini	Aggiungere una piccola quantità di rumore a un'immagine in modo che un sistema di riconoscimento facciale identifichi erroneamente una persona
Effetti Potenziali	Il modello apprende pattern errati e fa previsioni errate	Previsioni errate immediate e potenzialmente pericolose
Applicazioni Interessate	Qualsiasi modello ML	Veicoli autonomi, sistemi di sicurezza, ecc.

12.6.3 Inversione del Modello

Gli attacchi di inversione del modello rappresentano una minaccia per la privacy dei modelli di machine learning su dispositivo addestrati su dati utente sensibili (Nguyen et al. 2023). Comprendere questo vettore di attacco e le strategie di mitigazione saranno importanti per creare un'intelligenza artificiale su dispositivo sicura ed etica. Ad esempio, si immagini un'app per iPhone che utilizza l'apprendimento su dispositivo per categorizzare le foto in gruppi come "spiaggia", "cibo" o "selfie" per una ricerca più semplice.

Il modello su dispositivo potrebbe essere addestrato da Apple su un set di dati di foto iCloud di utenti consenzienti. Un aggressore malintenzionato potrebbe tentare di estrarre parti di quelle foto di addestramento iCloud originali utilizzando l'inversione del modello. In particolare, l'aggressore inserisce input sintetici creati ad arte nel classificatore di foto su dispositivo. Modificando gli input sintetici e osservando come il modello li categorizza, possono perfezionare gli input fino a ricostruire copie dei dati di training originali, come una foto di una spiaggia dall'iCloud di un utente. Ora, l'aggressore ha violato la privacy di quell'utente ottenendo una delle sue foto senza consenso. Questo dimostra perché l'inversione del modello è pericolosa: può potenzialmente far trapelare dati di training altamente sensibili.

Le foto sono un tipo di dati particolarmente rischioso perché spesso contengono persone identificabili, informazioni sulla posizione e momenti privati. Tuttavia, la stessa metodologia di attacco potrebbe essere applicata ad altri dati personali, come registrazioni audio, messaggi di testo o dati sanitari degli utenti.

Per difendersi dall'inversione del modello, sarebbe necessario prendere precauzioni come l'aggiunta di rumore agli output del modello o l'utilizzo di tecniche di apprendimento automatico che preservano la privacy come l'apprendimento federato per addestrare il modello sul dispositivo. L'obiettivo è impedire agli aggressori di ricostruire i dati di training originali.

12.6.4 Problemi di Sicurezza dell'Apprendimento On-Device

Sebbene l'avvelenamento dei dati e gli attacchi avversari siano preoccupazioni comuni per i modelli ML in generale, l'apprendimento su dispositivo introduce rischi di sicurezza unici. Quando vengono pubblicate varianti su dispositivo di modelli su larga scala, gli avversari possono sfruttare questi modelli più piccoli per attaccare le loro controparti più grandi. La

ricerca ha dimostrato che man mano che i modelli su dispositivo e i modelli su scala reale diventano più simili, la vulnerabilità dei modelli originali su larga scala aumenta in modo significativo. Ad esempio, le valutazioni su 19 reti neurali profonde (DNN) hanno rivelato che lo sfruttamento dei modelli su dispositivo potrebbe aumentare la vulnerabilità dei modelli originali su larga scala di fino a 100 volte.

Esistono tre tipi principali di rischi per la sicurezza specifici dell'apprendimento on-device:

- **Attacchi Basati sul Trasferimento:** Questi attacchi sfruttano la proprietà di trasferibilità tra un modello surrogato (un'approssimazione del modello di destinazione, simile a un modello su dispositivo) e un modello di destinazione remoto (il modello originale su scala reale). Gli aggressori generano esempi avversari utilizzando il modello surrogato, che può quindi essere utilizzato per ingannare il modello di destinazione. Ad esempio, si immagini un modello on-device progettato per identificare le e-mail di spam. Un aggressore potrebbe usare questo modello per generare un'e-mail di spam che non viene rilevata dal sistema di filtraggio più grande e completo.
- **Attacchi Basati sull'Ottimizzazione:** Questi attacchi generano esempi avversari per attacchi basati sul trasferimento usando una qualche forma di funzione obiettivo e modificano iterativamente gli input per ottenere il risultato desiderato. Gli attacchi di stima del gradiente, ad esempio, approssimano il gradiente del modello usando output di query (come punteggi di confidenza softmax), mentre gli attacchi senza gradiente usano la decisione finale del modello (la classe prevista) per approssimare il gradiente, sebbene richiedano molte più query.
- **Attacchi di Query con Priorità di Trasferimento:** Questi attacchi combinano elementi di attacchi basati sul trasferimento e basati sull'ottimizzazione. Eseguono il reverse engineering dei modelli sul dispositivo per fungere da surrogati del modello completo di destinazione. In altre parole, gli aggressori usano il modello sul dispositivo più piccolo per capire come funziona il modello più grande e quindi usano questa conoscenza per attaccare il modello completo.

Grazie alla comprensione di questi rischi specifici associati all'apprendimento on-device, possiamo sviluppare protocolli di sicurezza più solidi per proteggere sia i modelli on-device che quelli su scala reale da potenziali attacchi.

12.6.5 Attenuazione dei Rischi dell'Apprendimento On-Device

Si possono impiegare vari metodi per mitigare i numerosi rischi per la sicurezza associati all'apprendimento on-device. Questi metodi possono essere specifici per il tipo di attacco o fungere da strumento generale per rafforzare la sicurezza.

Una strategia per ridurre i rischi per la sicurezza è quella di ridurre la somiglianza tra modelli on-device e modelli su scala reale, riducendo così la trasferibilità fino al 90%. Questo metodo, noto come similarity-unpairing, affronta il problema che si verifica quando gli avversari sfruttano la somiglianza del gradiente di input tra i due modelli. Ottimizzando il modello su scala reale per creare una nuova versione con accuratezza simile ma gradienti di input diversi, possiamo costruire il modello on-device quantizzando questo modello su scala reale aggiornato. Questa disassociazione riduce la vulnerabilità dei modelli su dispositivo limitando l'esposizione del modello su scala reale originale. È importante notare che l'ordine di ottimizzazione e quantizzazione può essere variato pur ottenendo la mitigazione del rischio (Hong, Carlini, e Kurakin 2023).

Per contrastare l'avvelenamento dei dati, è fondamentale reperire set di dati da fornitori affidabili e fidati.

Per combattere gli attacchi avversari, si possono impiegare diverse strategie. Un approccio proattivo prevede la generazione di esempi avversari e la loro incorporazione nel set di dati di

training del modello, rafforzando così il modello contro tali attacchi. Strumenti come CleverHans, una libreria di training open source, sono fondamentali per creare esempi avversari. La "Defense distillation" [distillazione della difesa] è un'altra strategia efficace, in cui il modello sul dispositivo genera probabilità di classificazioni diverse anziché decisioni definitive (Hong, Carlini, e Kurakin 2023), rendendo più difficile per gli esempi avversari sfruttare il modello.

Il furto di proprietà intellettuale è un altro problema significativo quando si distribuiscono modelli on-device. Il furto di proprietà intellettuale è un problema quando si distribuiscono modelli on-device, poiché gli avversari potrebbero tentare di sottoporre a reverse engineering il modello per rubare la tecnologia sottostante. Per proteggersi dal furto di proprietà intellettuale, l'eseguibile binario del modello addestrato dovrebbe essere archiviato su un'unità microcontrollore con software crittografato e interfacce fisiche protette del chip. Inoltre, il set di dati finale utilizzato per l'addestramento del modello dovrebbe essere mantenuto privato.

Inoltre, i modelli on-device utilizzano spesso set di dati noti o open source, come Visual Wake Words di MobileNet. Pertanto, è importante mantenere la privacy del set di dati finale utilizzato per l'addestramento del modello. Inoltre, proteggere il processo di "data augmentation" e incorporare casi d'uso specifici può ridurre al minimo il rischio di reverse engineering di un modello on-device.

Infine, l'Adversarial Threat Landscape for Artificial Intelligence Systems (ATLAS) funge da prezioso strumento matriciale che aiuta a valutare il profilo di rischio dei modelli su dispositivo, consentendo agli sviluppatori di identificare e mitigare i potenziali rischi in modo proattivo.

12.6.6 Protezione dei Dati di Training

Esistono vari modi per proteggere i dati di training sul dispositivo. Ogni concetto è molto profondo e potrebbe valere una lezione a sé stante. Quindi, qui, faremo un breve accenno a quei concetti in modo che si sappia cosa approfondire.

12.6.6.1 Crittografia

La crittografia funge da prima linea di difesa per i dati di training. Ciò comporta l'implementazione della crittografia end-to-end per l'archiviazione locale su dispositivi e canali di comunicazione per impedire l'accesso non autorizzato ai dati di training grezzi. Ambienti di esecuzione affidabili, come Intel SGX e ARM TrustZone, sono essenziali per facilitare il training sicuro su dati crittografati.

Inoltre, quando si aggregano aggiornamenti da più dispositivi, è possibile impiegare protocolli di elaborazione "multi-party" sicuri per migliorare la sicurezza (Kairouz, Oh, e Viswanath 2015); un'applicazione pratica di ciò è nell'apprendimento collaborativo on-device, in cui è possibile implementare l'aggregazione crittografica che preserva la privacy degli aggiornamenti del modello utente. Questa tecnica nasconde efficacemente i dati dei singoli utenti anche durante la fase di aggregazione.

12.6.6.2 Privacy Differenziale

La privacy differenziale è un'altra strategia cruciale per proteggere i dati di training. Iniettando rumore statistico calibrato nei dati, possiamo mascherare i singoli record estraendo comunque preziosi pattern di popolazione (Dwork e Roth 2013). Anche la gestione del budget per la privacy su più iterazioni di training e la riduzione del rumore man mano che il modello converge sono essenziali (Abadi et al. 2016). Possono essere impiegati metodi come la privacy differenziale formalmente dimostrabile, che può includere l'aggiunta di rumore di Laplace o gaussiano scalato alla sensibilità del set di dati.

12.6.6.3 Rilevamento delle Anomalie

Il rilevamento delle anomalie svolge un ruolo importante nell'identificazione e nell'attenuazione di potenziali attacchi di avvelenamento dei dati. Ciò può essere ottenuto tramite analisi statistiche come la "Principal Component Analysis (PCA)" [analisi delle componenti principali] e il clustering, che aiutano a rilevare deviazioni nei dati di training aggregati. I metodi di serie temporali come i grafici Cumulative Sum (CUSUM) sono utili per identificare spostamenti indicativi di potenziale avvelenamento. Anche il confronto delle distribuzioni dei dati correnti con distribuzioni di dati pulite precedentemente visualizzate può aiutare a segnalare anomalie. Inoltre, i batch sospetti di essere avvelenati dovrebbero essere rimossi dal processo di aggregazione degli aggiornamenti di training. Ad esempio, è possibile condurre controlli a campione su sottoinsiemi di immagini di training sui dispositivi utilizzando hash photoDNA per identificare input avvelenati.

12.6.6.4 Validazione dei Dati di Input

Infine, la convalida dei dati di input è essenziale per garantire l'integrità e la validità dei dati di input prima che vengano immessi nel modello di training, proteggendo così dai payload avversari. Misure di similarità, come la distanza del coseno, possono essere impiegate per catturare input che si discostano in modo significativo dalla distribuzione prevista. Gli input sospetti che potrebbero contenere payload avversari devono essere messi in quarantena e sanificati. Inoltre, l'accesso del parser ai dati di training deve essere limitato solo ai percorsi di codice convalidati. Sfruttare le funzionalità di sicurezza hardware, come ARM Pointer Authentication, può impedire la corruzione della memoria (ARM Limited, 2023). Un esempio di ciò è l'implementazione di controlli di integrità degli input sui dati di training audio utilizzati dagli smart speaker prima dell'elaborazione da parte del modello di riconoscimento vocale (Zhiyong Chen e Xu 2023).

12.7 Framework di Training On-Device

Framework di inferenza embedded come TF-Lite Micro (David et al. 2021), TVM (T. Chen et al. 2018) e MCUNet (J. Lin et al. 2020) forniscono un runtime snello per l'esecuzione di modelli di reti neurali su microcontrollori e altri dispositivi con risorse limitate. Tuttavia, non supportano l'addestramento on-device. L'addestramento richiede un proprio set di strumenti specializzati a causa dell'impatto della quantizzazione sul calcolo del gradiente e dell'ingombro di memoria della backpropagation (J. Lin et al. 2022).
Negli ultimi anni, hanno iniziato a emergere una manciata di strumenti e framework che consentono l'addestramento sul dispositivo. Tra questi Tiny Training Engine (J. Lin et al. 2022), TinyTL (H. Cai et al. 2020) e TinyTrain (Y. D. Kwon et al. 2023).

12.7.1 Tiny Training Engine

Tiny Training Engine (TTE) utilizza diverse tecniche per ottimizzare l'utilizzo della memoria e velocizzare il processo di training. Una panoramica del flusso di lavoro TTE è mostrata in Figura 12.11. Innanzitutto, TTE scarica la differenziazione automatica in fase di compilazione anziché in fase di runtime, riducendo significativamente il sovraccarico durante il training. In secondo luogo, TTE esegue l'ottimizzazione del grafo come la potatura e gli aggiornamenti sparsi per ridurre i requisiti di memoria e accelerare i calcoli.

Figura 12.11: Flusso di lavoro di TTE.

In particolare, TTE segue quattro passaggi principali:

- Durante la fase di compilazione, TTE traccia il grafo di propagazione "forward" e deriva il grafo "backward" corrispondente per la backpropagation. Ciò consente alla differenziazione di avvenire in fase di compilazione anziché in fase di esecuzione.
- TTE elimina tutti i nodi che rappresentano pesi congelati dal grafo backward. I pesi congelati sono pesi che non vengono aggiornati durante l'addestramento per ridurre l'impatto di determinati neuroni. La potatura dei loro nodi consente di risparmiare memoria.
- TTE riordina gli operatori di discesa del gradiente per intercalarli con i calcoli del passaggio del backward. Questa pianificazione riduce al minimo le "impronte" [occupazione] di memoria.
- TTE utilizza la generazione di codice per compilare i grafi "forward" e "backward" ottimizzati, che vengono poi distribuiti per l'addestramento on-device.

12.7.2 Tiny Transfer Learning

Tiny Transfer Learning (TinyTL) consente un training efficiente in termini di memoria sul dispositivo tramite una tecnica chiamata congelamento dei pesi. Durante il training, gran parte del collo di bottiglia della memoria deriva dall'archiviazione delle attivazioni intermedie e dall'aggiornamento dei pesi nella rete neurale.
Per ridurre questo sovraccarico di memoria, TinyTL congela la maggior parte dei pesi in modo che non debbano essere aggiornati durante il training. Ciò elimina la necessità di archiviare le attivazioni intermedie per le parti congelate della rete. TinyTL ottimizza solo i termini di bias, che sono molto più piccoli dei pesi. Una panoramica del flusso di lavoro TinyTL è mostrata in Figura 12.12.

Figura 12.12: Flusso di lavoro di TinyTL. Fonte: H. Cai et al. (2020).)

I pesi di congelamento si applicano a layer completamente connessi, nonché a layer di normalizzazione e convoluzionali. Tuttavia, solo l'adattamento dei bias limita la capacità del modello di apprendere e adattarsi a nuovi dati.
Per aumentare l'adattabilità senza molta memoria aggiuntiva, TinyTL utilizza un piccolo modello di apprendimento residuo. Questo affina le mappe delle feature intermedie per

produrre output migliori, anche con pesi fissi. Il modello residuo introduce un overhead minimo, inferiore al 3,8% in più rispetto al modello di base.
Congelando la maggior parte dei pesi, TinyTL riduce significativamente l'utilizzo della memoria durante l'addestramento on-device. Il modello residuo consente quindi di adattarsi e apprendere in modo efficace per l'attività. L'approccio combinato fornisce un addestramento on-device efficiente in termini di memoria con un impatto minimo sulla precisione del modello.

12.7.3 Tiny Train

TinyTrain riduce significativamente il tempo necessario per l'addestramento sul dispositivo aggiornando selettivamente solo determinate parti del modello. Ciò avviene utilizzando una tecnica chiamata aggiornamento sparso adattivo all'attività, come mostrato in Figura 12.13.

Figura 12.13: Flusso di lavoro di TinyTrain. Fonte: Y. D. Kwon et al. (2023).

In base ai dati utente, alla memoria e al calcolo disponibili sul dispositivo, TinyTrain sceglie dinamicamente quali layer della rete neurale aggiornare durante l'addestramento. Questa selezione di layer è ottimizzata per ridurre l'utilizzo di calcolo e memoria mantenendo un'elevata accuratezza.
Più specificamente, TinyTrain esegue prima il pre-addestramento offline del modello. Durante il pre-addestramento, non solo addestra il modello sui dati dell'attività, ma anche il meta-addestramento del modello. Meta-addestramento significa addestrare il modello sui metadati relativi al processo di addestramento stesso. Questo meta-addestramento migliora la capacità del modello di adattarsi in modo accurato anche quando sono disponibili dati limitati per l'attività target.
Poi, durante la fase di adattamento online, quando il modello viene personalizzato sul dispositivo, TinyTrain esegue aggiornamenti adattivi sparsi all'attività. Utilizzando i criteri relativi alle capacità del dispositivo, seleziona solo determinati layer da aggiornare tramite backpropagation. I layer vengono scelti per bilanciare accuratezza, utilizzo della memoria e tempo di elaborazione.
Aggiornando in modo sparso i layer su misura per il dispositivo e l'attività, TinyTrain riduce significativamente il tempo di addestramento sul dispositivo e l'utilizzo delle risorse. Il meta-training offline migliora anche l'accuratezza quando si adatta a dati limitati. Insieme, questi metodi consentono un training on-device rapido, efficiente e accurato.

12.7.4 Confronto

Tabella 12.3 riassume le principali somiglianze e differenze tra i diversi framework.

Tabella 12.3: Confronto di framework per l'ottimizzazione del training on-device.

Framework	Somiglianze	Differenze
Tiny Training Engine	• Addestramento sul dispositivo • Ottimizza memoria e calcolo • Sfrutta potatura, sparsità, ecc.	• Traccia grafi forward & backward • Elimina i pesi congelati • Interlaccia backprop e gradienti • Generazione di codice
TinyTL	• Addestramento sul dispositivo • Ottimizza memoria e calcolo • Sfrutta congelamento, sparsità, ecc.	• Congela la maggior parte dei pesi • Adatta solo i bias • Utilizza il modello residuo
TinyTrain	• Addestramento sul dispositivo • Ottimizza memoria e calcolo • Sfrutta sparsità, ecc.	• Meta-addestramento nel pre-addestramento • Aggiornamento sparse adattivo alle attività • Aggiornamento selettivo dei layer

12.8 Conclusione

Il concetto di apprendimento on-device [su dispositivo] è sempre più importante per aumentare l'usabilità e la scalabilità di TinyML. Questo capitolo ha esplorato le complessità dell'apprendimento on-device, esplorandone vantaggi e limiti, strategie di adattamento, algoritmi e tecniche chiave correlate, implicazioni di sicurezza e framework di training on-device esistenti ed emergenti.

L'apprendimento su dispositivo è, senza dubbio, un paradigma rivoluzionario che porta con sé numerosi vantaggi per le distribuzioni ML embedded ed edge. Eseguendo il training direttamente sui dispositivi endpoint, si elimina la necessità di una connettività cloud continua, rendendolo particolarmente adatto per applicazioni IoT ed edge computing. Presenta vantaggi quali maggiore privacy, facilità di conformità ed efficienza delle risorse. Allo stesso tempo, l'apprendimento su on-device deve affrontare limitazioni legate a vincoli hardware, dimensioni dei dati limitate e ridotta accuratezza e generalizzazione del modello. Meccanismi quali la ridotta complessità del modello, tecniche di ottimizzazione e compressione dei dati e metodi di apprendimento correlati quali apprendimento tramite trasferimento e apprendimento federato consentono ai modelli di adattarsi per apprendere ed evolversi in base a vincoli di risorse, fungendo così da fondamento per un efficace ML sui dispositivi edge.

Le problematiche critiche di sicurezza nell'apprendimento su dispositivo evidenziate in questo capitolo, che vanno dall'avvelenamento dei dati e dagli attacchi avversari ai rischi specifici introdotti dall'apprendimento on-device, devono essere affrontate in carichi di lavoro reali affinché l'apprendimento su dispositivo sia un paradigma praticabile. Strategie di mitigazione efficaci, quali convalida dei dati, crittografia, privacy differenziale, rilevamento delle anomalie e convalida dei dati di input, sono fondamentali per salvaguardare i sistemi di apprendimento on-device da queste minacce.

L'emergere di framework di training specializzati on-device, come Tiny Training Engine, Tiny Transfer Learning e Tiny Train, offre strumenti pratici che consentono un training efficiente sui dispositivi. Questi framework impiegano varie tecniche per ottimizzare l'utilizzo della memoria, ridurre il sovraccarico computazionale e semplificare il processo di training on-device.

In conclusione, l'apprendimento on-device è in prima linea in TinyML, promettendo un futuro in cui i modelli possono acquisire autonomamente conoscenze e adattarsi ad ambienti

mutevoli su dispositivi edge. L'applicazione dell'apprendimento on-device ha il potenziale per rivoluzionare vari ambiti, tra cui sanità, IoT industriale e città intelligenti. Tuttavia, il potenziale trasformativo dell'apprendimento on-device deve essere bilanciato con misure di sicurezza robuste per proteggere da violazioni dei dati e minacce avversarie. L'adozione di framework di training on-device innovativi e l'implementazione di protocolli di sicurezza rigorosi sono passaggi chiave per sbloccare il pieno potenziale dell'apprendimento su dispositivo. Man mano che questa tecnologia continua a evolversi, promette di rendere i nostri dispositivi più intelligenti, più reattivi e meglio integrati nella nostra vita quotidiana.

12.9 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale al proprio ritmo. Incoraggiamo sia gli studenti che gli insegnanti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Intro to TensorFlow Lite (TFLite).
- TFLite Optimization and Quantization.
- TFLite Quantization-Aware Training.
- Trasferimento dell'Apprendimento:
  - Transfer Learning: with Visual Wake Words example.
  - On-device Training and Transfer Learning.
- Addestramento Distribuito:
  - Distributed Training.
  - Distributed Training.
- Monitoraggio Continuo:
  - Continuous Evaluation Challenges for TinyML.
  - Federated Learning Challenges.
  - Continuous Monitoring with Federated ML.
  - Continuous Monitoring Impact on MLOps.

Video

- Video 12.1
- Video 12.2

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 12.1
- Esercizio 12.2
- Esercizio 12.3

⚠ Laboratori

Oltre agli esercizi, offriamo anche una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

13. Operazioni di ML

DALL·E 3 Prompt: Crea un'illustrazione rettangolare ampia e dettagliata di un flusso di lavoro di IA. L'immagine dovrebbe mostrare il processo in sei fasi, con un flusso da sinistra a destra: 1. Raccolta dati, con individui diversi come sesso e discendenza che utilizzano una varietà di dispositivi come laptop, smartphone e sensori per raccogliere dati. 2. Elaborazione dati, che mostra un data center con server attivi e database con luci luminose. 3. Training del modello, rappresentata da uno schermo di computer con codice, diagrammi di reti neurali e indicatori di avanzamento. 4. Valutazione del modello, con persone che esaminano l'analisi dei dati su grandi monitor. 5. Distribuzione, in cui l'IA è integrata in robotica, app mobili e apparecchiature industriali. 6. Monitoraggio, che mostra professionisti che monitorano le metriche delle prestazioni dell'IA su dashboard per verificare l'accuratezza e la deriva del concetto nel tempo. Ogni fase dovrebbe essere contrassegnata in modo distinto e lo stile dovrebbe essere pulito, elegante e moderno con una combinazione di colori dinamica e informativa.

Questo capitolo esplora le pratiche e le architetture necessarie per sviluppare, distribuire e gestire efficacemente i modelli ML durante il loro intero ciclo di vita. Esaminiamo le varie fasi del processo ML, tra cui raccolta dati, addestramento del modello, valutazione, distribuzione e monitoraggio. Anche l'importanza dell'automazione, della collaborazione e del miglioramento continuo è un argomento di cui discutiamo. Mettiamo a confronto diversi ambienti per la distribuzione del modello ML, dai server cloud ai dispositivi edge embedded, e analizziamo i loro vincoli distinti. Mostriamo come personalizzare la progettazione e le operazioni del sistema ML attraverso esempi concreti per prestazioni del modello affidabili e ottimizzate in qualsiasi ambiente target. L'obiettivo è fornire ai lettori una comprensione completa della gestione del modello ML in modo che possano creare ed eseguire con successo applicazioni ML che forniscano valore in modo sostenibile.

Obiettivi dell'Apprendimento

- Capire cos'è MLOps e perché è necessario
- Imparare i pattern architetturali per MLOps tradizionali
- Confrontare MLOps tradizionali con MLOps embedded nel ciclo di vita ML
- Identificare i vincoli chiave degli ambienti embedded
- Imparare strategie per mitigare i problemi del ML embedded
- Esaminare casi di studio del mondo reale che dimostrano i principi MLOps embedded
- Apprezzare la necessità di approcci olistici tecnici e umani

13.1 Introduzione

Machine Learning Operations (MLOps) è un approccio sistematico che combina machine learning (ML), data science e ingegneria del software per automatizzare il ciclo di vita end-to-end di ML. Ciò include tutto, dalla preparazione dei dati e dal training del modello alla distribuzione e alla manutenzione. MLOps garantisce che i modelli ML siano sviluppati, distribuiti e mantenuti in modo efficiente ed efficace.

Cominciamo prendendo un caso di esempio generale (ad esempio, ML non edge). Prendiamo in considerazione un'azienda di "ride sharing" che desidera distribuire un modello di machine learning per prevedere la domanda dei passeggeri in tempo reale. Il team di data science impiega mesi per sviluppare un modello, ma quando è il momento di distribuirlo, si rende conto che deve essere compatibile con l'ambiente di produzione del team di ingegneria. La distribuzione del modello richiede la ricostruzione da zero, il che comporta settimane di lavoro aggiuntivo. È qui che entra in gioco MLOps.

Con MLOps, protocolli e strumenti, il modello sviluppato dal team di data science può essere distribuito e integrato senza problemi nell'ambiente di produzione. In sostanza, MLOps elimina gli attriti durante lo sviluppo, la distribuzione e la manutenzione dei sistemi ML. Migliora la collaborazione tra i team tramite flussi di lavoro e interfacce definiti. MLOps accelera anche la velocità di iterazione consentendo la distribuzione continua per i modelli ML.

Per l'azienda di ride sharing, implementare MLOps significa che il loro modello di previsione della domanda può essere frequentemente riqualificato e distribuito in base ai nuovi dati in arrivo. Ciò mantiene il modello accurato nonostante il cambiamento del comportamento del passeggero. MLOps consente inoltre all'azienda di sperimentare nuove tecniche di modellazione poiché i modelli possono essere rapidamente testati e aggiornati.

Altri vantaggi di MLOps includono il monitoraggio avanzato della discendenza del modello, la riproducibilità e l'auditing. La catalogazione dei flussi di lavoro ML e la standardizzazione degli artefatti, come il logging delle versioni del modello, il monitoraggio della discendenza dei dati e il confezionamento di modelli e parametri, consente una visione più approfondita della provenienza del modello. La standardizzazione di questi artefatti facilita la tracciabilità di un modello fino alle sue origini, la replica del processo di sviluppo del modello e l'esame di come una versione del modello è cambiata nel tempo. Ciò facilita anche la conformità alle normative, che è particolarmente critica in settori regolamentati come sanità e finanza, dove è importante essere in grado di verificare e spiegare i modelli.

Le principali organizzazioni adottano MLOps per aumentare la produttività, aumentare la collaborazione e accelerare i risultati ML. Fornisce i framework, gli strumenti e le best practice per gestire efficacemente i sistemi ML durante il loro ciclo di vita. Ciò si traduce in modelli più performanti, tempi di realizzazione più rapidi e un vantaggio competitivo duraturo.

Mentre esploriamo ulteriormente MLOps, si consideri come l'implementazione di queste pratiche può aiutare ad affrontare le sfide ML embedded oggi e in futuro.

13.2 Contesto Storico

MLOps affonda le sue radici in DevOps, un insieme di pratiche che combinano sviluppo software (Dev) e operazioni IT (Ops) per accorciare il ciclo di vita dello sviluppo e fornire una distribuzione "continua" di software di alta qualità. I parallelismi tra MLOps e DevOps sono evidenti nella loro attenzione all'automazione, alla collaborazione e al miglioramento continuo. In entrambi i casi, l'obiettivo è quello di abbattere i "silos" tra i diversi team (sviluppatori, operazioni e, nel caso di MLOps, data scientist e ingegneri ML) e creare un processo più snello ed efficiente. È utile comprendere meglio la storia di questa evoluzione per comprendere MLOps nel contesto dei sistemi tradizionali.

13.2.1 DevOps

Il termine "DevOps" è stato coniato per la prima volta nel 2009 da Patrick Debois, un consulente e professionista Agile. Debois ha organizzato la prima conferenza DevOpsDays a Ghent, in Belgio, nel 2009. La conferenza ha riunito professionisti dello sviluppo e delle operazioni per discutere di modi per migliorare la collaborazione e automatizzare i processi.
DevOps ha le sue radici nel movimento Agile, iniziato nei primi anni 2000. Agile ha fornito le basi per un approccio più collaborativo allo sviluppo software e ha enfatizzato le piccole release iterative. Tuttavia, Agile si concentra principalmente sulla collaborazione tra team di sviluppo. Man mano che le metodologie Agile diventavano più popolari, le organizzazioni si sono rese conto della necessità di estendere questa collaborazione ai team operativi.
La natura isolata dei team di sviluppo e delle operazioni ha spesso portato a inefficienze, conflitti e ritardi nella distribuzione del software. Questa necessità di una migliore collaborazione e integrazione tra questi team ha portato al movimento DevOps. DevOps può essere visto come un'estensione dei principi Agile, inclusi i team operativi.
I principi chiave di DevOps includono collaborazione, automazione, integrazione continua, distribuzione e feedback. DevOps si concentra sull'automazione dell'intera pipeline di distribuzione del software, dallo sviluppo alla distribuzione. Migliora la collaborazione tra i team di sviluppo e operativi, utilizzando strumenti come Jenkins, Docker e Kubernetes per semplificare il ciclo di vita dello sviluppo.
Mentre Agile e DevOps condividono principi comuni in materia di collaborazione e feedback, DevOps mira specificamente all'integrazione di sviluppo e operazioni IT, espandendo Agile oltre i soli team di sviluppo. Introduce pratiche e strumenti per automatizzare la distribuzione del software e migliorare la velocità e la qualità delle release del software.

13.2.2 MLOps

MLOps, d'altro canto, sta per Machine Learning Operations ed estende i principi di DevOps al ciclo di vita ML. MLOps automatizza e semplifica l'intero ciclo di vita dell'apprendimento automatico, dalla preparazione dei dati allo sviluppo del modello, fino all'implementazione e al monitoraggio. L'obiettivo principale di MLOps è facilitare la collaborazione tra data scientist, data engineer e operazioni IT e automatizzare la distribuzione, il monitoraggio e la gestione dei modelli ML. Alcuni fattori chiave hanno portato all'ascesa di MLOps.

- **Data drift:** La deriva dei dati degrada le prestazioni del modello nel tempo, motivando la necessità di rigorosi monitoraggi e procedure di riqualificazione automatizzate fornite da MLOps.
- **Riproducibilità:** La mancanza di riproducibilità negli esperimenti di machine learning ha motivato i sistemi MLOps a tracciare codice, dati e variabili di ambiente per abilitare flussi di lavoro ML riproducibili.

- **Spiegabilità:** La natura di "scatola nera" e la mancanza di spiegabilità di modelli complessi hanno motivato la necessità di funzionalità MLOps per aumentare la trasparenza e la spiegabilità del modello.
- **Monitoraggio:** L'incapacità di monitorare in modo affidabile le prestazioni del modello dopo la distribuzione ha evidenziato la necessità di soluzioni MLOps con una solida strumentazione delle prestazioni del modello e avvisi.
- **Attrito:** L'attrito nel riaddestramento e nella distribuzione manuale dei modelli ha motivato la necessità di sistemi MLOps che automatizzano le pipeline di distribuzione dell'apprendimento automatico.
- **Ottimizzazione:** La complessità della configurazione dell'infrastruttura di apprendimento automatico ha motivato la necessità di piattaforme MLOps con un'infrastruttura ML ottimizzata e pronta all'uso.

Sebbene DevOps e MLOps condividano l'obiettivo comune di automatizzare e semplificare i processi, differiscono significativamente in termini di attenzione e sfide. DevOps si occupa principalmente di sviluppo software e operazioni IT. Consente la collaborazione tra questi team e automatizza la distribuzione del software. Al contrario, MLOps si concentra sul ciclo di vita dell'apprendimento automatico. Affronta complessità aggiuntive come versioning dei dati, versioning dei modelli e monitoraggio dei modelli. MLOps richiede la collaborazione tra una gamma più ampia di stakeholder, tra cui data scientist, data engineer e IT operations. Va oltre l'ambito del DevOps tradizionale incorporando le sfide uniche della gestione dei modelli ML durante il loro ciclo di vita. Tabella 13.1 fornisce un confronto affiancato di DevOps e MLOps, evidenziandone le principali differenze e somiglianze.

Tabella 13.1: Confronto tra DevOps e MLOps.

Aspect	DevOps	MLOps
Obiettivo	Semplificazione dei processi di sviluppo software e operativi	Ottimizzazione del ciclo di vita dei modelli di apprendimento automatico
Metodologia	Integrazione continua e distribuzione continua (CI/CD) per lo sviluppo software	Simile a CI/CD ma incentrato sui flussi di lavoro di apprendimento automatico
Strumenti Principali	Controllo delle versioni (Git), strumenti CI/CD (Jenkins, Travis CI), gestione della configurazione (Ansible, Puppet)	Strumenti di versioning dei dati, strumenti di training e deployment dei modelli, pipeline CI/CD su misura per ML
Problemi Principali	Integrazione del codice, test, gestione delle release, automazione, infrastruttura come codice	Gestione dei dati, versioning dei modelli, monitoraggio degli esperimenti, deployment dei modelli, scalabilità dei flussi di lavoro ML
Risultati Tipici	Release software più rapide e affidabili, collaborazione migliorata tra team di sviluppo e operativi	Gestione e deployment efficienti dei modelli di apprendimento automatico, collaborazione migliorata tra data scientist e ingegneri

Scoprire di più sui cicli di vita ML tramite un "case study" che presenta il riconoscimento vocale in Video 13.1.

Video 13.1: MLOps

https://www.youtube.com/watch?v=YJsRD_hU4tc&list=PLkDaE6sCZn6GMoA0wbpJLi3t34Gd8l0aK&i

13.3 Componenti Chiave di MLOps

In questo capitolo, forniremo una panoramica dei componenti principali di MLOps, un set emergente di pratiche che consente una distribuzione solida e una gestione del ciclo di vita dei modelli ML in produzione. Sebbene alcuni elementi MLOps, come l'automazione e il monitoraggio, siano stati trattati nei capitoli precedenti, li integreremo in un framework e

approfondiremo funzionalità aggiuntive, come la governance. Inoltre, descriveremo e collegheremo gli strumenti più diffusi utilizzati in ogni componente, come LabelStudio per l'etichettatura dei dati. Alla fine, speriamo che abbiate compreso la metodologia MLOps end-to-end che porta i modelli dall'ideazione alla creazione di valore sostenibile all'interno delle organizzazioni.
Figura 13.1 illustra lo stack completo del sistema MLOps. Mostra i vari layer coinvolti nelle operazioni di apprendimento automatico. In cima allo stack ci sono modelli/applicazioni ML, come BERT, seguiti da framework/piattaforme ML come PyTorch. Il livello MLOps principale, etichettato come Model Orchestration, comprende diversi componenti chiave: Data Management, CI/CD, Model Training, Model Evaluation, Deployment e Model Serving. Alla base del livello MLOps c'è il livello Infrastructure, rappresentato da tecnologie come Kubernetes. Questo livello gestisce aspetti come Job Scheduling, Resource Management, Capacity Management e Monitoring, tra gli altri. A tenere tutto insieme c'è il layer Hardware, che fornisce le risorse computazionali necessarie per le operazioni ML.

Figura 13.1: Lo stack MLOps, inclusi Modelli ML, Framework, Orchestrazione del Modello, Infrastruttura e Hardware, illustra il flusso di lavoro end-to-end di MLOps.

Questo approccio a "layer" [strati] in Figura 13.1 dimostra come MLOps integra varie tecnologie e processi per facilitare lo sviluppo, la distribuzione e la gestione di modelli di apprendimento automatico in un ambiente di produzione. La figura illustra efficacemente le interdipendenze tra diversi componenti e come si uniscono per formare un ecosistema MLOps completo.

13.3.1 Gestione dei Dati

Una gestione dei dati solida e l'ingegneria dei dati potenziano attivamente implementazioni MLOps di successo. I team acquisiscono, archiviano e preparano correttamente i dati grezzi da sensori, database, app e altri sistemi per il training e la distribuzione dei modelli.
I team monitorano attivamente le modifiche ai set di dati nel tempo utilizzando il controllo delle versioni con Git e strumenti come GitHub o GitLab. Gli scienziati dei dati collaborano alla cura dei set di dati unendo le modifiche di più collaboratori. I team possono rivedere o ripristinare ogni iterazione di un set di dati, se necessario.

I team etichettano e annotano meticolosamente i dati utilizzando un software di etichettatura come LabelStudio, che consente ai team distribuiti di lavorare insieme all'etichettatura dei set di dati. Man mano che le variabili target e le convenzioni di etichettatura si evolvono, i team mantengono l'accessibilità alle versioni precedenti.
I team archiviano il set di dati non elaborato e tutte le risorse derivate su servizi cloud come Amazon S3 o Google Cloud Storage. Questi servizi forniscono un'archiviazione scalabile e resiliente con funzionalità di controllo delle versioni. I team possono impostare autorizzazioni di accesso granulari.
Le pipeline di dati robuste create dai team automatizzano l'estrazione, l'unione, la pulizia e la trasformazione dei dati grezzi in set di dati pronti per l'analisi. Prefect, Apache Airflow e dbt sono orchestratori di flussi di lavoro che consentono agli ingegneri di sviluppare pipeline di elaborazione dati flessibili e riutilizzabili.
Ad esempio, una pipeline può acquisire dati da database PostgreSQL, API REST e CSV archiviati su S3 [Simple Storage Service]. Può filtrare, deduplicare e aggregare i dati, gestire gli errori e salvare l'output su S3. La pipeline può anche spingere i dati trasformati in un feature store come Tecton o Feast per un accesso a bassa latenza.
In un caso d'uso di manutenzione predittiva industriale, i dati dei sensori vengono acquisiti dai dispositivi in S3. Una pipeline perfetta elabora i dati dei sensori, unendoli ai record di manutenzione. Il set di dati arricchito è archiviato in Feast in modo che i modelli possano recuperare facilmente i dati più recenti per l'addestramento e le previsioni.
Video 13.2 di seguito riporta una breve panoramica delle pipeline di dati.

Video 13.2: Pipeline di Dati

https://www.youtube.com/watch?v=gz-44N3MMOA&list=PLkDaE6sCZn6GMoA0wbpJLi3t34Gd8l0aK&index=33

13.3.2 Pipeline CI/CD

Le pipeline di integrazione continua e distribuzione continua (CI/CD) automatizzano attivamente la progressione dei modelli ML dallo sviluppo iniziale alla distribuzione in produzione. Adattati per i sistemi ML, i principi CI/CD consentono ai team di distribuire rapidamente e in modo robusto nuovi modelli con errori manuali ridotti al minimo.
Le pipeline CI/CD orchestrano i passaggi chiave, tra cui il controllo delle nuove modifiche al codice, la trasformazione dei dati, il training e la registrazione di nuovi modelli, i test di convalida, la containerizzazione, la distribuzione in ambienti come cluster di staging e la promozione in produzione. I team sfruttano le soluzioni CI/CD più diffuse come Jenkins, CircleCI e GitHub Actions per eseguire queste pipeline MLOps, mentre Prefect, Metaflow e Kubeflow offrono opzioni incentrate su ML.
Figura 13.2 illustra una pipeline CI/CD specificamente pensata per MLOps. Il processo inizia con un dataset e un repository di feature (a sinistra), che alimenta una fase di ingestione del dataset. Dopo l'ingestione, i dati vengono sottoposti a convalida per garantirne la qualità prima di essere trasformati per l'addestramento. Parallelamente, un trigger di riaddestramento può avviare la pipeline in base a criteri specificati. I dati passano poi attraverso una fase di addestramento/ottimizzazione del modello all'interno di un motore di elaborazione dati, seguita dalla valutazione e convalida del modello. Una volta convalidato, il modello viene registrato e archiviato in un repository di metadati e artefatti di apprendimento automatico. La fase finale prevede la distribuzione del modello addestrato nuovamente nel dataset e nel repository di feature, creando così un processo ciclico per il miglioramento continuo e la distribuzione di modelli di apprendimento automatico.

Figura 13.2: Diagramma CI/CD MLOps. Fonte: HarvardX.

Ad esempio, quando uno scienziato dei dati verifica i miglioramenti a un modello di classificazione delle immagini in un repository GitHub, questo attiva attivamente una pipeline CI/CD Jenkins. La pipeline riesegue le trasformazioni dei dati e l'addestramento del modello sui dati più recenti, monitorando gli esperimenti con MLflow. Dopo i test di validazione automatizzati, i team distribuiscono il contenitore del modello in un cluster di staging Kubernetes per un ulteriore controllo qualità. Una volta approvato, Jenkins facilita un rollout graduale del modello in produzione con distribuzioni canary per rilevare eventuali problemi. Se vengono rilevate anomalie, la pipeline consente ai team di tornare alla versione precedente del modello in modo fluido.
Le pipeline CI/CD consentono ai team di iterare e distribuire rapidamente modelli ML collegando i diversi passaggi dallo sviluppo alla distribuzione con automazione continua. L'integrazione di strumenti MLOps come MLflow migliora il packaging del modello, il controllo delle versioni e la tracciabilità della pipeline. CI/CD è fondamentale per far progredire i modelli oltre i prototipi in sistemi aziendali sostenibili.

13.3.3 Addestramento del Modello

Nella fase di training del modello, gli scienziati dei dati sperimentano attivamente diverse architetture e algoritmi ML per creare modelli ottimizzati che estraggono informazioni e pattern dai dati. MLOps introduce best practice e automazione per rendere questo processo iterativo più efficiente e riproducibile.
I moderni framework ML come TensorFlow, PyTorch e Keras forniscono componenti predefiniti che semplificano la progettazione di reti neurali e altre architetture di modelli. Gli scienziati dei dati sfruttano moduli integrati per layer, attivazioni, perdite, ecc. e API di alto livello come Keras per concentrarsi maggiormente sull'architettura del modello.
MLOps consente ai team di impacchettare il codice di training del modello in script e notebook riutilizzabili e tracciati. Man mano che i modelli vengono sviluppati, funzionalità come ottimizzazione degli iperparametri, ricerca dell'architettura neurale e selezione automatica delle funzionalità si ripetono rapidamente per trovare le configurazioni più performanti.
I team utilizzano Git per controllare le versioni del codice di training e ospitarlo in repository come GitHub per tenere traccia delle modifiche nel tempo. Ciò consente una collaborazione fluida tra i data scientist.
Notebook come Jupyter creano un eccellente ambiente di sviluppo di modelli interattivi. I notebook contengono l'inserimento dei dati, la pre-elaborazione, la dichiarazione del

modello, il ciclo di training, la valutazione e il codice di esportazione in un documento riproducibile.
Infine, i team orchestrano il training del modello come parte di una pipeline CI/CD per l'automazione. Ad esempio, una pipeline Jenkins può attivare uno script Python per caricare nuovi dati di training, riaddestrare un classificatore TensorFlow, valutare le metriche del modello e registrare automaticamente il modello se vengono raggiunte le soglie di prestazione.
Un esempio di flusso di lavoro prevede che uno scienziato dei dati utilizzi un notebook PyTorch per sviluppare un modello CNN per la classificazione delle immagini. La libreria fastai fornisce API di alto livello per semplificare l'addestramento delle CNN sui set di dati delle immagini. Il notebook addestra il modello sui dati campione, valuta le metriche di accuratezza e ottimizza gli iperparametri come la velocità di apprendimento e i layer per ottimizzare le prestazioni. Questo notebook riproducibile è controllato dalla versione e integrato in una pipeline di riaddestramento.
L'automazione e la standardizzazione dell'addestramento del modello consentono ai team di accelerare la sperimentazione e raggiungere il rigore necessario per produrre sistemi ML.

13.3.4 Valutazione del Modello

Prima di distribuire i modelli, i team eseguono una valutazione e dei test rigorosi per convalidare i benchmark delle prestazioni e la prontezza per il rilascio. MLOps introduce le best practice relative alla convalida, all'audit e ai test canary dei modelli.
In genere, i team valutano i modelli rispetto ai dataset di test di holdout che non vengono utilizzati durante il training. I dati di test provengono dalla stessa distribuzione dei dati di produzione. I team calcolano metriche come accuratezza, AUC, precisione, richiamo e punteggio F1.
I team monitorano inoltre le stesse metriche nel tempo rispetto ai campioni di dati di test. Se i dati di valutazione provengono da flussi di produzione live, questo rileva le derive dei dati che degradano le prestazioni del modello nel tempo.
La supervisione umana per il rilascio del modello rimane importante. Gli scienziati dei dati esaminano le prestazioni nei segmenti e nelle sezioni chiave. L'analisi degli errori aiuta a identificare i punti deboli del modello per guidare il miglioramento. team applicano tecniche di equità e rilevamento di bias.
Il test Canary rilascia un modello a un piccolo sottoinsieme di utenti per valutare le prestazioni nel mondo reale prima di un'ampia distribuzione. I team indirizzano gradualmente il traffico alla versione canary monitorando i problemi.
Ad esempio, un rivenditore valuta un modello di raccomandazione di prodotto personalizzato rispetto ai dati di test storici, esaminando le metriche di accuratezza e diversità. I team calcolano anche le metriche sui dati dei clienti in tempo reale nel tempo, rilevando una riduzione dell'accuratezza nelle ultime 2 settimane. Prima dell'implementazione completa, il nuovo modello viene rilasciato al 5% del traffico web per garantire che non vi sia alcun degrado.
L'automazione della valutazione e delle versioni canary riduce i rischi di distribuzione. Tuttavia, la revisione umana deve ancora essere più critica per valutare le dinamiche meno quantificabili del comportamento del modello. Una rigorosa convalida pre-distribuzione fornisce sicurezza nell'immissione dei modelli in produzione.

13.3.5 Distribuzione del Modello

I team devono confezionare, testare e tracciare correttamente i modelli ML per distribuirli in modo affidabile in produzione. MLOps introduce framework e procedure per il versioning attivo, la distribuzione, il monitoraggio e l'aggiornamento dei modelli in modi sostenibili.

I team containerizzano i modelli utilizzando Docker, che raggruppa codice, librerie e dipendenze in un'unità standardizzata. I container consentono una portabilità fluida tra gli ambienti.
Framework come TensorFlow Serving e BentoML aiutano a servire le previsioni dai modelli distribuiti tramite API ottimizzate per le prestazioni. Questi framework gestiscono il versioning, il ridimensionamento e il monitoraggio.
I team distribuiscono prima i modelli aggiornati in ambienti di staging o QA per i test prima del rollout completo in produzione. Le distribuzioni shadow o canary instradano un campione di traffico per testare le varianti del modello. I team aumentano gradualmente l'accesso ai nuovi modelli.
I team creano solide procedure di rollback nel caso in cui emergano problemi. I rollback ripristinano l'ultima versione valida del modello. L'integrazione con pipeline CI/CD semplifica la ridistribuzione, se necessario.
I team monitorano attentamente gli artefatti del modello, come script, pesi, log e metriche, per ogni versione con strumenti di metadati ML come MLflow. Ciò mantiene la discendenza e la verificabilità.
Ad esempio, un rivenditore inserisce in un "contenitore" un modello di raccomandazione di prodotto in TensorFlow Serving e lo distribuisce in un cluster di staging Kubernetes. Dopo aver monitorato e approvato le prestazioni sul traffico di esempio, Kubernetes sposta il 10% del traffico di produzione al nuovo modello. Se non vengono rilevati problemi dopo alcuni giorni, il nuovo modello occupa il 100% del traffico. Tuttavia, i team dovrebbero mantenere la versione precedente accessibile per il rollback, se necessario.
I processi di distribuzione del modello consentono ai team di rendere i sistemi ML resilienti in produzione tenendo conto di tutti gli stati di transizione.

13.3.6 Model Serving

Dopo il "deployment" distribuzione del modello, ML-as-a-Service diventa un componente fondamentale nel ciclo di vita di MLOps. I servizi online come Facebook/Meta gestiscono decine di trilioni di query di inferenza al giorno (C.-J. Wu et al. 2019). Il "model serving" colma il divario tra i modelli sviluppati e le applicazioni ML o gli utenti finali, assicurando che i modelli distribuiti siano accessibili, performanti e scalabili negli ambienti di produzione.
Diversi framework facilitano il model serving, tra cui TensorFlow Serving, NVIDIA Triton Inference Server e KServe (in precedenza KFServing). Questi strumenti forniscono interfacce standardizzate per la distribuzione di modelli distribuiti su varie piattaforme e gestiscono molte complessità dell'inferenza del modello su larga scala.
Il model serving può essere categorizzato in tre tipi principali:

1. **Online Serving:** Fornisce previsioni in tempo reale con bassa latenza, il che è fondamentale per applicazioni come sistemi di raccomandazione o rilevamento frodi.
2. **Offline Serving:** Elabora grandi batch di dati in modo asincrono, adatto per attività come la generazione periodica di report.
3. **Near-Online Serving (semi-sincrono):** Bilancia tra online e offline, offrendo risposte relativamente rapide per applicazioni meno sensibili al tempo come i chatbot.

Una delle sfide principali per i sistemi di model serving è operare secondo requisiti di prestazioni definiti da Service Level Agreement (SLA) e Service Level Objective (SLO). Gli SLA sono contratti formali che specificano i livelli di servizio previsti. Questi livelli di servizio si basano su parametri quali tempo di risposta, disponibilità e produttività. Gli SLO sono obiettivi interni che i team si prefiggono di soddisfare o superare i propri SLA.
Per il model serving ML, gli accordi e gli obiettivi SLA e SLO hanno un impatto diretto sull'esperienza utente, sull'affidabilità del sistema e sui risultati aziendali. Pertanto, i team

ottimizzano attentamente la propria piattaforma di servizio. I serving system ML impiegano varie tecniche per ottimizzare le prestazioni e l'utilizzo delle risorse, come le seguenti:

1. **Pianificazione e batch delle richieste:** Gestisce in modo efficiente le richieste di inferenza ML in arrivo, ottimizzando le prestazioni tramite strategie di accodamento e raggruppamento intelligenti. Sistemi come Clipper (Crankshaw et al. 2017) introducono il servizio di previsione online a bassa latenza con tecniche di caching e batch.
2. **Selezione e routing delle istanze del modello:** Algoritmi intelligenti indirizzano le richieste alle versioni o alle istanze del modello appropriate. INFaaS (Romero et al. 2021) esplora questo aspetto generando varianti del modello e navigando in modo efficiente nello spazio di compromesso in base ai requisiti di prestazioni e accuratezza.
3. **Bilanciamento del carico:** Distribuisce i carichi di lavoro in modo uniforme su più istanze di servizio. MArk (Model Ark) (Chengliang Zhang et al. 2019) dimostra tecniche efficaci di bilanciamento del carico per sistemi di servizio ML.
4. **Autoscaling delle istanze del modello:** Regola dinamicamente la capacità in base alla domanda. Sia INFaaS (Romero et al. 2021) che MArk (Chengliang Zhang et al. 2019) incorporano funzionalità di autoscaling per gestire in modo efficiente le fluttuazioni del carico di lavoro.
5. **Orchestration del modello:** Gestisce l'esecuzione del modello, abilitando l'elaborazione parallela e l'allocazione strategica delle risorse. AlpaServe (Zhuohan Li et al. 2023) dimostra tecniche avanzate per la gestione di modelli di grandi dimensioni e scenari di servizio complessi.
6. **Previsione del tempo di esecuzione:** Sistemi come Clockwork (Gujarati et al. 2020) si concentrano sul servizio ad alte prestazioni prevedendo i tempi di esecuzione delle singole inferenze e utilizzando in modo efficiente gli acceleratori hardware.

I serving system ML che eccellono in queste aree consentono alle organizzazioni di distribuire modelli che funzionano in modo affidabile sotto pressione. Il risultato sono applicazioni AI scalabili e reattive in grado di gestire le richieste del mondo reale e fornire valore in modo coerente.

13.3.7 Gestione dell'Infrastruttura

I team MLOps sfruttano ampiamente gli strumenti "infrastructure as code (IaC)" e le solide architetture cloud per gestire attivamente le risorse necessarie per lo sviluppo, il training e la distribuzione dei sistemi ML.

I team utilizzano strumenti IaC come Terraform, CloudFormation e Ansible per definire, fornire e aggiornare a livello di programmazione l'infrastruttura in modo controllato dalla versione. Per MLOps, i team utilizzano ampiamente Terraform per avviare risorse su AWS, GCP e Azure.

Per la creazione e il training dei modelli, i team forniscono dinamicamente risorse di elaborazione come server GPU, cluster di container, storage e database tramite Terraform in base alle esigenze degli scienziati dei dati. Il codice incapsula e preserva le definizioni dell'infrastruttura.

I container e gli orchestratori come Docker e Kubernetes consentono ai team di impacchettare modelli e distribuirli in modo affidabile in diversi ambienti. I contenitori possono essere attivati o disattivati automaticamente in base alla domanda.

Sfruttando l'elasticità del cloud, i team aumentano o diminuiscono le risorse per soddisfare i picchi nei carichi di lavoro come i lavori di ottimizzazione degli iperparametri o i picchi nelle richieste di previsione. Auto-scaling consente un'efficienza dei costi ottimizzata.

L'infrastruttura si estende su dispositivi on-prem, cloud ed edge. Uno stack tecnologico robusto offre flessibilità e resilienza. Gli strumenti di monitoraggio consentono ai team di osservare l'utilizzo delle risorse.
Ad esempio, una configurazione Terraform può distribuire un cluster GCP Kubernetes per ospitare modelli TensorFlow addestrati esposti come microservizi di previsione. Il cluster aumenta i pod per gestire un traffico maggiore. L'integrazione CI/CD distribuisce senza problemi nuovi contenitori di modelli.
La gestione attenta dell'infrastruttura tramite IaC e monitoraggio consente ai team di prevenire i colli di bottiglia nell'operatività dei sistemi ML su larga scala.

13.3.8 Monitoraggio

I team MLOps mantengono attivamente un monitoraggio robusto per mantenere la visibilità nei modelli ML distribuiti in produzione. Il monitoraggio continuo fornisce informazioni sulle prestazioni del modello e del sistema in modo che i team possano rilevare e risolvere rapidamente i problemi per ridurre al minimo le interruzioni.
I team monitorano attivamente gli aspetti chiave del modello, inclusa l'analisi di campioni di previsioni live per tracciare metriche come accuratezza e matrice di confusione nel tempo. Quando monitorano le prestazioni, i team devono profilare i dati in arrivo per verificare la deriva del modello, un calo costante dell'accuratezza del modello dopo l'implementazione in produzione. La deriva del modello può verificarsi in due modi: deriva del concetto e deriva dei dati. La deriva del concetto si riferisce a un cambiamento fondamentale osservato nella relazione tra i dati di input e quelli target. Ad esempio, con l'avanzare della pandemia di COVID-19, i siti di e-commerce e vendita al dettaglio hanno dovuto correggere le raccomandazioni del modello poiché i dati di acquisto erano ampiamente distorti verso articoli come il disinfettante per le mani. La deriva dei dati descrive i cambiamenti nella distribuzione dei dati nel tempo. Ad esempio, gli algoritmi di riconoscimento delle immagini utilizzati nelle auto a guida autonoma devono tenere conto della stagionalità nell'osservazione dell'ambiente circostante. I team monitorano anche le metriche delle prestazioni delle applicazioni come latenza ed errori per le integrazioni dei modelli.
Da una prospettiva infrastrutturale, i team monitorano i problemi di capacità come elevato utilizzo di CPU, memoria e disco e interruzioni del sistema. Strumenti come Prometheus, Grafana ed Elastic consentono ai team di raccogliere, analizzare, interrogare e visualizzare attivamente diverse metriche di monitoraggio. Le dashboard rendono le dinamiche altamente visibili.
I team configurano gli allarmi per le metriche di monitoraggio chiave come cali di accuratezza e guasti del sistema per consentire una risposta proattiva agli eventi che minacciano l'affidabilità. Ad esempio, i cali di accuratezza del modello attivano avvisi per i team per esaminare potenziali deviazioni dei dati e riaddestrare i modelli utilizzando campioni di dati aggiornati e rappresentativi.
Dopo la distribuzione, il monitoraggio completo consente ai team di mantenere la fiducia nello stato del modello e del sistema. Consente ai team di rilevare e risolvere preventivamente le deviazioni tramite allarmi e dashboard basati sui dati. Il monitoraggio attivo è essenziale per mantenere sistemi ML altamente disponibili e affidabili.
Guardare il video qui sotto per saperne di più sul monitoraggio.

Video 13.3: Monitoraggio del Modello

https://www.youtube.com/watch?v=hq_XyP9y0xg&list=PLkDaE6sCZn6GMoA0wbpJLi3t34Gd8l0aK&index=7

13.3.9 Governance

I team MLOps stabiliscono attivamente pratiche di governance appropriate come componente fondamentale. La governance fornisce una supervisione sui modelli ML per garantire che siano affidabili, etici e conformi. Senza governance, sussistono rischi significativi di modelli che si comportano in modi pericolosi o proibiti quando vengono distribuiti in applicazioni e processi aziendali.

La governance MLOps impiega tecniche per fornire trasparenza sulle previsioni, sulle prestazioni e sul comportamento del modello durante l'intero ciclo di vita ML. Metodi di spiegabilità come SHAP e LIME aiutano gli auditor a comprendere perché i modelli effettuano determinate previsioni evidenziando le caratteristiche di input influenti alla base delle decisioni. Bias detection analizza le prestazioni del modello in diversi gruppi demografici definiti da attributi come età, sesso ed etnia per rilevare eventuali distorsioni sistematiche. I team eseguono rigorose procedure di test su set di dati rappresentativi per convalidare le prestazioni del modello prima della distribuzione.

Una volta in produzione, i team monitorano la concept drift [deriva del concetto] per determinare se le relazioni predittive cambiano nel tempo in modi che degradano l'accuratezza del modello. I team analizzano anche i registri di produzione per scoprire pattern nei tipi di errori generati dai modelli. La documentazione sulla provenienza dei dati, le procedure di sviluppo e le metriche di valutazione fornisce ulteriore visibilità.

Piattaforme come Watson OpenScale incorporano funzionalità di governance come il monitoraggio dei bias e la spiegabilità direttamente nella creazione di modelli, nei test e nel monitoraggio della produzione. Le aree di interesse principali della governance sono trasparenza, correttezza e conformità. Ciò riduce al minimo i rischi che i modelli si comportino in modo errato o pericoloso quando integrati nei processi aziendali.

L'integrazione di pratiche di governance nei flussi di lavoro MLOps consente ai team di garantire un'IA affidabile.

13.3.10 Comunicazione e Collaborazione

MLOps abbatte attivamente i "silos" e consente il libero flusso di informazioni e approfondimenti tra i team in tutte le fasi del ciclo di vita ML. Strumenti come MLflow, Weights & Biases e contesti di dati forniscono tracciabilità e visibilità per migliorare la collaborazione.

I team utilizzano MLflow per sistematizzare il monitoraggio di esperimenti, versioni e artefatti del modello. Gli esperimenti possono essere loggati a livello di programmazione da notebook di data science e job di training. Il registro dei modelli fornisce un hub centrale per i team per archiviare modelli pronti per la produzione prima della distribuzione, con metadati come descrizioni, metriche, tag e discendenza. Le integrazioni con Github, GitLab facilitano i trigger per la modifica del codice.

"Weights & Biases" fornisce strumenti collaborativi su misura per i team ML. Gli scienziati dei dati registrano gli esperimenti, visualizzano metriche come curve di perdita e condividono approfondimenti sulla sperimentazione con i colleghi. Le dashboard di confronto evidenziano le differenze del modello. I team discutono dei progressi e dei passaggi successivi.

La definizione di contesti di dati condivisi, ovvero glossari, dizionari di dati e riferimenti di schemi, garantisce l'allineamento del significato e dell'utilizzo dei dati tra i ruoli. La documentazione aiuta a comprendere chi non ha accesso diretto ai dati.

Ad esempio, uno scienziato dei dati può utilizzare "Weights & Biases" per analizzare un esperimento con un modello di rilevamento delle anomalie e condividere i risultati della valutazione con altri membri del team per discutere dei miglioramenti. Il modello finale può quindi essere registrato con MLflow prima di essere consegnato per la distribuzione.

L'abilitazione della trasparenza, della tracciabilità e della comunicazione tramite MLOps consente ai team di rimuovere i colli di bottiglia e accelerare la distribuzione di sistemi ML di impatto.
Video 13.4 affronta le sfide chiave nella distribuzione del modello, tra cui la deriva del concetto, la deriva del modello e i problemi di ingegneria del software.

Video 13.4: Sfide della Distribuzione

https://www.youtube.com/watch?v=UyEtTyeahus&list=PLkDaE6sCZn6GMoA0wbpJLi3t34Gd8l0aK&index=5

13.4 Debito Tecnico Nascosto nei Sistemi ML

Il debito tecnico [https://it.wikipedia.org/wiki/Debito_tecnico] è sempre più pressante per i sistemi di apprendimento automatico. Questa metafora, originariamente proposta negli anni '90, paragona i costi a lungo termine dello sviluppo rapido del software al debito finanziario. Proprio come un debito finanziario alimenta una crescita vantaggiosa, un debito tecnico gestito con attenzione consente una rapida iterazione. Tuttavia, se non controllato, l'accumulo di debito tecnico può superare qualsiasi guadagno.
Figura 13.3 illustra i vari componenti che contribuiscono al debito tecnico nascosto dei sistemi ML. Mostra la natura interconnessa di configurazione, raccolta dati ed estrazione di funzionalità, che è fondamentale per la base di codice ML. Le dimensioni delle caselle indicano la proporzione dell'intero sistema rappresentata da ciascun componente. Nei sistemi ML industriali, il codice per l'algoritmo del modello costituisce solo una piccola frazione (vedere la piccola casella nera al centro rispetto a tutte le altre caselle grandi). La complessità dei sistemi ML e la natura frenetica del settore rendono molto facile l'accumulo di debito tecnico.

Figura 13.3: Componenti del sistema ML. Fonte: Sambasivan et al. (2021b)

13.4.1 Erosione dei Confini del Modello

A differenza del software tradizionale, ML non ha confini chiari tra i componenti, come si vede nel diagramma sopra. Questa erosione dell'astrazione crea intrecci che esacerbano il debito tecnico in diversi modi:

13.4.2 Intreccio

Un accoppiamento stretto tra i componenti del modello ML rende difficile isolare le modifiche. La modifica di una parte provoca effetti a catena imprevedibili in tutto il sistema. "Changing Anything Changes Everything (noto anche come CACE)" [Cambiare qualcosa cambia tutto] è un fenomeno che si applica a qualsiasi modifica apportata al sistema. Le potenziali mitigazioni includono la scomposizione del problema quando possibile o il monitoraggio ravvicinato delle modifiche nel comportamento per contenerne l'impatto.

13.4.3 Cascate di Correzione

Figura 13.4 illustra il concetto di cascate di correzione nel flusso di lavoro ML, dalla definizione del problema all'implementazione del modello. Gli archi rappresentano le potenziali correzioni iterative necessarie in ogni fase del flusso di lavoro, con colori diversi corrispondenti a problemi distinti come l'interazione con la fragilità del mondo fisico, competenze inadeguate nel dominio dell'applicazione, sistemi di ricompensa in conflitto e scarsa documentazione inter-organizzativa.
Le frecce rosse indicano l'impatto delle cascate, che possono portare a revisioni significative nel processo di sviluppo del modello. Al contrario, la linea rossa tratteggiata rappresenta la misura drastica di abbandono del processo per riavviarlo. Questa immagine sottolinea la natura complessa e interconnessa dello sviluppo del sistema ML e l'importanza di affrontare questi problemi all'inizio del ciclo di sviluppo per mitigare i loro effetti di amplificazione a valle.

Figura 13.4: Diagramma di flusso delle cascate di correzione. Fonte: Sambasivan et al. (2021b).

La creazione di modelli in sequenza crea dipendenze rischiose in cui i modelli successivi si basano su quelli precedenti. Ad esempio, prendere un modello esistente e perfezionarlo per un nuovo caso d'uso sembra efficiente. Tuttavia, questo incorpora ipotesi dal modello originale che potrebbero eventualmente richiedere una correzione.
Diversi fattori influenzano la decisione di creare modelli in sequenza o meno:

- **Dimensioni del dataset e tasso di crescita:** Con set di dati statici e di piccole dimensioni, la messa a punto dei modelli esistenti ha spesso senso. Per set di dati di grandi dimensioni e in crescita, l'addestramento di modelli personalizzati da zero consente una maggiore flessibilità per tenere conto dei nuovi dati.
- **Risorse di elaborazione disponibili:** La messa a punto richiede meno risorse rispetto all'addestramento di modelli di grandi dimensioni da zero. Con risorse limitate, sfruttare i modelli esistenti potrebbe essere l'unico approccio fattibile.

Mentre la messa a punto dei modelli esistenti può essere efficiente, la modifica dei componenti fondamentali in seguito diventa estremamente costosa a causa di questi effetti a cascata. Pertanto, si dovrebbe considerare attentamente l'introduzione di nuove architetture di modelli, anche se ad alta intensità di risorse, per evitare cascate di correzioni in futuro. Questo approccio può aiutare ad attenuare gli effetti di amplificazione dei problemi a valle e a ridurre il debito tecnico. Tuttavia, ci sono ancora scenari in cui la creazione di modelli sequenziali ha senso, il che richiede un attento equilibrio tra efficienza, flessibilità e manutenibilità a lungo termine nel processo di sviluppo ML.

13.4.4 Consumatori Non Dichiarati

Una volta che le previsioni del modello ML sono rese disponibili, molti sistemi downstream [derivati] potrebbero utilizzarle silenziosamente come input per un'ulteriore elaborazione. Tuttavia, il modello originale non è stato progettato per adattarsi a questo ampio riutilizzo. A causa dell'opacità intrinseca dei sistemi ML, diventa impossibile analizzare completamente

l'impatto degli output del modello come input altrove. Le modifiche al modello possono quindi avere conseguenze costose e pericolose interrompendo dipendenze non rilevate.
I "consumatori" non dichiarati possono anche abilitare loop di feedback nascosti se i loro output influenzano indirettamente i dati di training del modello originale. Le mitigazioni includono la limitazione dell'accesso alle previsioni, la definizione di contratti di servizio rigorosi e il monitoraggio di segnali di influenze non-modellate. Architettare sistemi ML per incapsulare e isolare i loro effetti limita i rischi di propagazione imprevista.

13.4.5 Debito di Dipendenza dai Dati

Il debito di dipendenza dei dati si riferisce a dipendenze di dati instabili e sottoutilizzate, che possono avere ripercussioni dannose e difficili da rilevare. Sebbene questo sia un fattore chiave del debito tecnologico per il software tradizionale, tali sistemi possono trarre vantaggio dall'uso di strumenti ampiamente disponibili per l'analisi statica da parte di compilatori e linker per identificare dipendenze di questo tipo. I sistemi ML necessitano di strumenti simili.
Una mitigazione per le dipendenze di dati instabili è l'uso del versioning, che garantisce la stabilità degli input ma comporta il costo della gestione di più set di dati e il potenziale della obsolescenza. Un'altra mitigazione per le dipendenze di dati sottoutilizzate è quella di condurre una valutazione esaustiva "leave-one-feature-out".

13.4.6 Debito di Analisi dai Cicli di Feedback

A differenza del software tradizionale, i sistemi ML possono cambiare il loro comportamento nel tempo, rendendo difficile l'analisi pre-distribuzione. Questo debito si manifesta nei cicli di feedback, sia diretti che nascosti.
I cicli di feedback diretti si verificano quando un modello influenza i suoi input futuri, ad esempio consigliando prodotti agli utenti che, a loro volta, modellano i dati di training futuri. I cicli nascosti sorgono indirettamente tra modelli, ad esempio due sistemi che interagiscono tramite ambienti del mondo reale. I cicli di feedback graduali sono particolarmente difficili da rilevare. Questi cicli portano al debito di analisi, ovvero l'incapacità di prevedere come un modello agirà completamente dopo il rilascio. Essi compromettono la validazione pre-distribuzione consentendo un'autoinfluenza non modellata.
Un attento monitoraggio e distribuzioni "canary" aiutano a rilevare il feedback. Tuttavia, permangono sfide fondamentali nella comprensione delle interazioni complesse del modello. Le scelte architettoniche che riducono l'intreccio e l'accoppiamento mitigano l'effetto composto del debito di analisi.

13.4.7 Le Giungle di Pipeline

I workflow [flussi di lavoro] ML spesso necessitano di interfacce più standardizzate tra i componenti. Ciò porta i team a "incollare" gradualmente le pipeline con codice personalizzato. Ciò che emerge sono "giungle di pipeline", ovvero passaggi di pre-elaborazione aggrovigliati che sono fragili e resistono al cambiamento. Evitare modifiche a queste pipeline disordinate fa sì che i team sperimentino attraverso prototipi alternativi. Presto, proliferano molteplici modi di fare. La necessità di astrazioni e interfacce impedisce quindi la condivisione, il riutilizzo e l'efficienza.
Il debito tecnico si accumula man mano che le pipeline si solidificano in vincoli legacy. I team sprecano tempo nella gestione di codice idiosincratico anziché massimizzare le prestazioni del modello. Principi architettonici come modularità e incapsulamento sono necessari per stabilire interfacce pulite. Le astrazioni condivise consentono componenti intercambiabili, impediscono il lock-in e promuovono la diffusione delle "best practice" tra i team. Liberarsi dalle "giungle di pipeline" richiede in definitiva l'applicazione di standard che impediscano

l'accumulo di debito di astrazione. I vantaggi delle interfacce e delle API che domano la complessità superano i costi di transizione.

13.4.8 Debito di Configurazione

I sistemi ML comportano una configurazione estesa di iperparametri, architetture e altri parametri di ottimizzazione. Tuttavia, la configurazione è spesso un ripensamento, che necessita di più rigore e test: aumentano le configurazioni ad hoc, amplificate dalle numerose "manopole" disponibili per l'ottimizzazione di modelli ML complessi.
Questo accumulo di debito tecnico ha diverse conseguenze. Configurazioni fragili e obsolete portano a dipendenze nascoste e bug che causano guasti di produzione. La conoscenza sulle configurazioni ottimali è isolata anziché condivisa, portando a un lavoro ridondante. Riprodurre e confrontare i risultati diventa difficile quando le configurazioni mancano di documentazione. I vincoli legacy si accumulano poiché i team temono di modificare configurazioni poco comprese.
Per affrontare il debito di configurazione è necessario stabilire standard per documentare, testare, convalidare e archiviare centralmente le configurazioni. Investire in approcci più automatizzati, come l'ottimizzazione degli iperparametri e la ricerca dell'architettura, riduce la dipendenza dall'ottimizzazione manuale. Una migliore igiene della configurazione rende il miglioramento iterativo più gestibile impedendo alla complessità di aumentare all'infinito. La chiave è riconoscere la configurazione come parte integrante del ciclo di vita del sistema ML piuttosto che come un ripensamento ad hoc.

13.4.9 Il Mondo che Cambia

I sistemi ML operano in ambienti dinamici del mondo reale. Le soglie e le decisioni inizialmente efficaci diventano obsolete man mano che il mondo si evolve. Tuttavia, i vincoli legacy rendono difficile adattare i sistemi a popolazioni, pattern di utilizzo e altri fattori contestuali mutevoli.
Questo debito si manifesta in due modi principali. In primo luogo, le soglie preimpostate e le euristiche richiedono una rivalutazione e una messa a punto costanti man mano che i loro valori ottimali si spostano. In secondo luogo, la convalida dei sistemi tramite test statici di unità e integrazione fallisce quando input e comportamenti sono obiettivi in movimento. Rispondere a un mondo in continua evoluzione in tempo reale con sistemi ML legacy è impegnativo. Il debito tecnico si accumula man mano che le ipotesi decadono. La mancanza di architettura modulare e la capacità di aggiornare dinamicamente i componenti senza effetti collaterali esacerbano questi problemi.
Per mitigare questo problema è necessario integrare configurabilità, monitoraggio e aggiornabilità modulare. L'apprendimento online, in cui i modelli si adattano continuamente e solidi cicli di feedback alle pipeline di training, aiutano a sintonizzarsi automaticamente sul mondo. Tuttavia, anticipare e progettare il cambiamento è essenziale per prevenire l'erosione delle prestazioni nel mondo reale nel tempo.

13.4.10 Gestire il Debito Tecnico nelle Fasi Iniziali

È comprensibile che il debito tecnico si accumuli naturalmente nelle prime fasi di sviluppo del modello. Quando si punta a creare rapidamente modelli MVP, i team spesso hanno bisogno di informazioni più complete su quali componenti raggiungeranno la scala o richiederanno modifiche. È previsto un po' di lavoro differito.
Tuttavia, anche i sistemi iniziali frammentati dovrebbero seguire principi come "Flexible Foundations" per evitare di mettersi nei guai:

- Il codice modulare e le librerie riutilizzabili consentono di scambiare i componenti in un secondo momento

- L'accoppiamento debole tra modelli, archivi dati e logica aziendale facilita il cambiamento
- I layer di astrazione nascondono i dettagli di implementazione che potrebbero cambiare nel tempo
- Il servizio di modelli containerizzati mantiene aperte le opzioni sui requisiti di distribuzione

Le decisioni che sembrano ragionevoli al momento possono limitare seriamente la flessibilità futura. Ad esempio, incorporare la logica aziendale chiave nel codice modello anziché tenerla separata rende estremamente difficili le modifiche successive al modello.
Con una progettazione ponderata, tuttavia, è possibile creare rapidamente all'inizio mantenendo gradi di libertà per migliorare. Man mano che il sistema matura, emergono prudenti punti di interruzione in cui l'introduzione di nuove architetture in modo proattivo evita massicce rilavorazioni in futuro. In questo modo si bilanciano le urgenti tempistiche con la riduzione delle future cascate di correzione.

13.4.11 Riepilogo

Sebbene il debito finanziario sia una buona metafora per comprendere i compromessi, differisce dalla misurabilità del debito tecnico. Il debito tecnico deve essere completamente monitorato e quantificato. Ciò rende difficile per i team gestire i compromessi tra muoversi rapidamente e introdurre intrinsecamente più debito rispetto al prendersi il tempo per ripagare tale debito.
Il documento Hidden Technical Debt of Machine Learning Systems diffonde la consapevolezza delle sfumature del debito tecnologico specifico del sistema ML. Incoraggia un ulteriore sviluppo nell'ampia area del ML manutenibile.

13.5 Ruoli e Responsabilità

Data la vastità di MLOps, l'implementazione di successo di sistemi ML richiede competenze diversificate e una stretta collaborazione tra persone con diverse aree di competenza. Mentre gli scienziati dei dati creano i modelli ML di base, è necessario un lavoro di squadra interfunzionale per distribuire con successo questi modelli in ambienti di produzione e consentire loro di fornire un valore aziendale sostenibile.
MLOps fornisce il framework e le pratiche per coordinare gli sforzi di vari ruoli coinvolti nello sviluppo, nella distribuzione e nell'esecuzione di sistemi MLG. Collegare i "silos" tradizionali tra i team di dati, ingegneria e operazioni è fondamentale per il successo di MLOps. Abilitare una collaborazione senza soluzione di continuità attraverso il ciclo di vita dell'apprendimento automatico accelera la realizzazione dei vantaggi garantendo al contempo l'affidabilità e le prestazioni a lungo termine dei modelli ML.
Esamineremo alcuni ruoli chiave coinvolti in MLOps e le loro responsabilità principali. Comprendere l'ampiezza delle competenze necessarie per rendere operativi i modelli ML guida l'assemblaggio dei team MLOps. Chiarisce inoltre come i flussi di lavoro tra i ruoli si adattano alla metodologia MLOps sovraordinata.

13.5.1 Ingegneri dei Dati

Gli ingegneri dei dati sono responsabili della creazione e della manutenzione dell'infrastruttura dati e delle pipeline che alimentano i dati nei modelli ML. Garantiscono che i dati vengano trasferiti senza problemi dai sistemi di origine agli ambienti di archiviazione, elaborazione e progettazione delle funzionalità necessari per lo sviluppo e la distribuzione dei modelli ML. Le loro principali responsabilità includono:

- Migrare dati grezzi da database, sensori e app “on-prem” [in azienda], in data lake basati su cloud, come Amazon S3 o Google Cloud Storage. Ciò fornisce un’archiviazione economica e scalabile.
- Creare pipeline di dati con “scheduler” [pianificatori] di flussi di lavoro come Apache Airflow, Prefect e dbt. Questi estraggono i dati dalle sorgenti, li trasformano e li convalidano, e li caricano direttamente in destinazioni come data warehouse, feature store o per l’addestramento del modello.
- Trasformare dati grezzi e disordinati in set di dati strutturati e pronti per l’analisi. Ciò include la gestione di valori nulli o malformati, la deduplicazione, l’unione di origini dati disparate, l’aggregazione dei dati e la progettazione di nuove feature.
- Manutenere componenti dell’infrastruttura dati come data warehouse cloud (Snowflake, Redshift, BigQuery), data lake e sistemi di gestione dei metadati. Provisioning e ottimizzazione dei sistemi di elaborazione dati.
- Fornire e ottimizzare sistemi di elaborazione dati per una gestione e un’analisi dei dati efficiente e scalabile.
- Definire i processi di versioning, backup e archiviazione dei dati per i set di dati e funzionalità ML e applicare policy di governance dei dati.

Ad esempio, un’azienda manifatturiera può utilizzare pipeline Apache Airflow per estrarre dati dei sensori dai PLC in fabbrica e trasferirli in un data lake Amazon S3. Gli ingegneri dei dati elaborerebbero poi questi dati grezzi per filtrarli, pulirli e unirli ai metadati del prodotto. Questi output della pipeline verrebbero quindi caricati in un data warehouse Snowflake da cui è possibile leggere le feature per l’addestramento e la previsione del modello.
Il team di ingegneria dei dati crea e sostiene la base dati per uno sviluppo e un funzionamento affidabili del modello. Il loro lavoro consente agli scienziati dei dati e agli ingegneri ML di concentrarsi sulla creazione, l’addestramento e l’implementazione di modelli ML su larga scala.

13.5.2 Data Scientist

Il lavoro dei “data scientist” [scienziato dei dati] è concentrarsi sulla ricerca, sperimentazione, sviluppo e miglioramento continuo dei modelli ML. Sfruttano la loro competenza in statistica, modellazione e algoritmi per creare modelli ad alte prestazioni. Le loro principali responsabilità includono:

- Collaborare con team aziendali e di dati per identificare opportunità in cui ML può aggiungere valore, inquadrare il problema e definire metriche di successo.
- Eseguire analisi esplorative dei dati per comprendere le relazioni nei dati, ricavare informazioni e identificare funzionalità rilevanti per la modellazione.
- Ricercare e sperimentare diversi algoritmi ML e architetture di modelli in base al problema e alle caratteristiche dei dati e sfruttare librerie come TensorFlow, PyTorch e Keras.
- Massimizzare le prestazioni, addestrare e perfezionare i modelli regolando gli iperparametri, regolando le architetture delle reti neurali, l’ingegneria delle funzionalità, ecc.
- Valutare le prestazioni del modello tramite metriche come accuratezza, AUC e punteggi F1 ed eseguire analisi degli errori per identificare aree di miglioramento.
- Sviluppare nuove versioni del modello mediante l’integrazione di nuovi dati, test di diversi approcci, ottimizzazione del comportamento del modello e mantenimento della documentazione e della discendenza per i modelli.

Ad esempio, uno scienziato dei dati può sfruttare TensorFlow e TensorFlow Probability per sviluppare un modello di previsione della domanda per la pianificazione dell'inventario ala vendita al dettaglio. Itereranno su diversi modelli di sequenza come LSTM e sperimenteranno funzionalità derivate da dati di prodotto, vendite e stagionali. Il modello verrà valutato in base a metriche di errore rispetto alla domanda effettiva prima dell'implementazione. Lo scienziato dei dati monitora le prestazioni e riqualifica/migliora il modello man mano che arrivano nuovi dati.
I data scientist guidano la creazione, il miglioramento e l'innovazione del modello attraverso la loro competenza nelle tecniche di ML. Collaborano strettamente con altri ruoli per garantire che i modelli creino il massimo impatto aziendale.

13.5.3 ML Engineer

Gli "ingegneri ML" consentono ai modelli sviluppati dagli scienziati dei dati di essere prodotti e distribuiti su larga scala. La loro competenza fa sì che i modelli servano in modo affidabile alle previsioni nelle applicazioni e nei processi aziendali. Le loro principali responsabilità includono:

- Prendere modelli prototipo dagli scienziati dei dati e rafforzarli per gli ambienti di produzione tramite best practice di codifica.
- Creare API e microservizi per la distribuzione dei modelli utilizzando strumenti come Flask, FastAPI. Containerizzare i modelli con Docker.
- Gestire le versioni dei modelli, sincronizzarli in produzione utilizzando pipeline CI/CD e implementare release canary, test A/B e procedure di rollback.
- Ottimizzare le prestazioni dei modelli per elevata scalabilità, bassa latenza ed efficienza dei costi. Sfruttare compressione, quantizzazione e servizio multi-modello.
- Monitorare i modelli una volta in produzione e garantire affidabilità e precisione continue. Riqualificare periodicamente i modelli.

Ad esempio, un ingegnere ML può prendere un modello di rilevamento delle frodi TensorFlow sviluppato da data scientist e containerizzarlo utilizzando TensorFlow Serving per una distribuzione scalabile. Il modello verrebbe integrato nella pipeline di elaborazione delle transazioni dell'azienda tramite API. L'ingegnere ML implementa un registro dei modelli e una pipeline CI/CD utilizzando MLFlow e Jenkins per distribuire gli aggiornamenti del modello in modo affidabile. Gli ingegneri ML monitorano quindi il modello in esecuzione per prestazioni continue utilizzando strumenti come Prometheus e Grafana. Se l'accuratezza del modello diminuisce, avviano la riqualificazione e la distribuzione di una nuova versione del modello.
Il team di ingegneria ML consente ai modelli di data science di progredire senza problemi in sistemi di produzione sostenibili e robusti. La loro competenza nella creazione di sistemi modulari e monitorati offre un valore aziendale continuo.

13.5.4 DevOps Engineer

Gli "ingegneri DevOps" abilitano MLOps creando e gestendo l'infrastruttura sottostante per lo sviluppo, la distribuzione e il monitoraggio dei modelli ML. Forniscono l'architettura cloud e le pipeline di automazione. Le loro principali responsabilità includono:

- Approvvigionare e gestire l'infrastruttura cloud per i flussi di lavoro ML utilizzando strumenti IaC come Terraform, Docker e Kubernetes.
- Sviluppare pipeline CI/CD per il riaddestramento, la convalida e la distribuzione del modello. Integrare strumenti ML nella pipeline, come MLflow e Kubeflow.
- Monitorare le prestazioni del modello e dell'infrastruttura tramite strumenti come Prometheus, Grafana, stack ELK. Creare allarmi e dashboard.

- Implementare pratiche di governance relative allo sviluppo, al test e alla promozione del modello per consentire riproducibilità e tracciabilità.
- Embedding dei modelli ML nelle applicazioni. Espongono i modelli tramite API e microservizi per l'integrazione.
- Ottimizzazione delle prestazioni e dei costi dell'infrastruttura e sfruttamento dell'autoscaling, delle istanze spot e della disponibilità in tutte le regioni.

Ad esempio, un ingegnere DevOps esegue il provisioning di un cluster Kubernetes su AWS utilizzando Terraform per eseguire lavori di training ML e distribuzione online. L'ingegnere crea una pipeline CI/CD in Jenkins, che attiva il riaddestramento del modello quando sono disponibili nuovi dati. Dopo il test automatizzato, il modello viene registrato con MLflow e distribuito nel cluster Kubernetes. L'ingegnere monitora quindi lo stato del cluster, l'utilizzo delle risorse del contenitore e la latenza dell'API utilizzando Prometheus e Grafana.
Il team DevOps consente una rapida sperimentazione e distribuzioni affidabili per ML tramite competenze cloud, automazione e monitoraggio. Il loro lavoro massimizza l'impatto del modello riducendo al minimo il debito tecnico.

13.5.5 Project Manager

I project manager svolgono un ruolo fondamentale in MLOps coordinando le attività tra i team coinvolti nella distribuzione dei progetti ML. Aiutano a guidare l'allineamento, la "accountability" [affidabilità] ed accelerano i risultati. Le loro principali responsabilità includono:

- Collaborare con le parti interessate per definire obiettivi di progetto, metriche di successo, tempistiche e budget; delineare specifiche e "scope".
- Creare un piano di progetto che comprenda acquisizione dati, sviluppo modello, configurazione infrastrutturale, distribuzione e monitoraggio.
- Coordinare i lavori di progettazione, sviluppo e test tra ingegneri dei dati, scienziati dei dati, ingegneri ML e ruoli DevOps.
- Monitorare i progressi e le milestone, identificare gli ostacoli e risolverli tramite azioni correttive e gestire rischi e problemi.
- Facilitare la comunicazione tramite report di stato, riunioni, workshop e documentazione e consentire una collaborazione senza interruzioni.
- Guidare l'aderenza alle tempistiche e al budget e aumentare i superamenti o le carenze previsti per la mitigazione.

Ad esempio, un project manager creerebbe un piano di progetto per sviluppare e migliorare un modello di previsione dell'abbandono dei clienti. Coordinare data engineer che creano pipeline di dati, data scientist che sperimentano modelli, ML engineer che producono modelli e DevOps che impostano l'infrastruttura di distribuzione. Il project manager monitora i progressi tramite milestone come preparazione del set di dati, prototipazione del modello, distribuzione e monitoraggio. Per attuare soluzioni preventive, evidenziano eventuali rischi, ritardi o problemi di budget.
I project manager qualificati consentono ai team MLOps di lavorare in sinergia per fornire rapidamente il massimo valore aziendale dagli investimenti ML. La loro leadership e organizzazione si allineano con team diversi.

13.6 Sfide dei Sistemi Embedded

Sulla base della nostra discussione su L'apprendimento "On-device" nel capitolo precedente, ora rivolgiamo la nostra attenzione al contesto più ampio dei sistemi embedded in MLOps. I vincoli e i requisiti unici degli ambienti embedded hanno un impatto significativo sull'implementazione di modelli e operazioni di apprendimento automatico. Per preparare il

terreno per le sfide specifiche che emergono con MLOps embedded, è importante esaminare prima le sfide generali associate ai sistemi embedded. Questa panoramica fornirà una base per comprendere come questi vincoli si intersecano e modellano le pratiche di MLOps in scenari di edge computing con risorse limitate.

13.6.1 Risorse di Elaborazione Limitate

I dispositivi embedded come i microcontrollori e i telefoni cellulari hanno una potenza di elaborazione molto più limitata rispetto alle macchine dei data center o alle GPU. Un tipico microcontrollore può avere solo KB di RAM, velocità della CPU MHz e nessuna GPU. Ad esempio, un microcontrollore in uno smartwatch può avere solo un processore a 32 bit in esecuzione a 120 MHz con 320 KB di RAM («EuroSoil 2021 (O205)» 2021). Ciò consente modelli ML semplici come piccole regressioni lineari o foreste casuali, ma reti neurali profonde più complesse sarebbero irrealizzabili. Le strategie per mitigare questo includono quantizzazione, potatura, architetture di modelli efficienti e scaricamento di determinati calcoli sul cloud quando la connettività lo consente.

13.6.2 Memoria Limitata

Memorizzare grandi modelli ML e set di dati direttamente su dispositivi embedded è spesso impossibile con una memoria limitata. Ad esempio, un modello di rete neurale profonda può facilmente occupare centinaia di MB, il che supera la capacità di archiviazione di molti sistemi embedded. Si consideri questo esempio. Una fotocamera per la fauna selvatica che cattura immagini per rilevare animali può avere solo una scheda di memoria da 2 GB. Ne serve di più per memorizzare un modello di deep learning per la classificazione delle immagini che spesso ha una dimensione di centinaia di MB. Di conseguenza, ciò richiede l'ottimizzazione dell'utilizzo della memoria tramite compressione dei pesi, numeri di precisione inferiore e pipeline di inferenza in streaming.

13.6.3 Connettività Intermittente

Molti dispositivi embedded operano in ambienti remoti senza una connettività Internet affidabile. Dobbiamo fare affidamento su qualcosa di diverso dall'accesso cloud costante per un comodo riaddestramento, monitoraggio e distribuzione. Al contrario, abbiamo bisogno di strategie intelligenti sia di pianificazione che si memorizzazione nella cache per ottimizzare le connessioni intermittenti. Ad esempio, un modello che prevede la resa del raccolto in una fattoria remota potrebbe dover fare previsioni giornaliere ma avere connettività al cloud solo una volta alla settimana quando l'agricoltore si reca in città. Il modello deve funzionare in modo indipendente tra una connessione e l'altra.

13.6.4 Limitazioni della Potenza

I dispositivi embedded come telefoni, dispositivi indossabili e sensori remoti sono alimentati a batteria. L'inferenza e la comunicazione continue possono esaurire rapidamente le batterie, limitandone la funzionalità. Ad esempio, un collare intelligente che contrassegna gli animali in via di estinzione funziona con una piccola batteria. L'esecuzione continua di un modello di tracciamento GPS scaricherebbe la batteria nel giro di pochi giorni. Il collare deve pianificare con attenzione quando attivare il modello. Pertanto, l'ML integrato deve gestire attentamente le attività per risparmiare energia. Le tecniche includono acceleratori hardware ottimizzati, caching delle previsioni ed esecuzione di modelli adattivi.

13.6.5 Gestione della Flotta

Per i dispositivi embedded prodotti in serie, milioni di unità possono essere distribuite sul campo per orchestrare gli aggiornamenti. Ipoteticamente, l'aggiornamento di un modello di rilevamento delle frodi su 100 milioni di carte di credito (future intelligenti) richiede l'invio sicuro degli aggiornamenti a ciascun dispositivo distribuito anziché a un data center

centralizzato. Una scala così distribuita rende la gestione dell'intera flotta molto più difficile di un cluster di server centralizzato. Richiede protocolli intelligenti per aggiornamenti "over-the-air", gestione dei problemi di connettività e monitoraggio dei vincoli di risorse tra i dispositivi.

13.6.6 Raccolta Dati On-Device

La raccolta di dati di training utili richiede la progettazione sia dei sensori sul dispositivo sia delle pipeline software. Questo è diverso dai server, dove possiamo estrarre dati da fonti esterne. Le sfide includono la gestione del rumore dei sensori. I sensori su una macchina industriale rilevano vibrazioni e temperatura per prevedere le esigenze di manutenzione. Ciò richiede la messa a punto dei sensori e delle frequenze di campionamento per acquisire dati utili.

13.6.7 Personalizzazione Specifica del Dispositivo

Uno smart speaker impara i pattern vocali e la cadenza del parlato di un singolo utente per migliorare la precisione del riconoscimento proteggendo al contempo la privacy. Adattare i modelli ML a dispositivi e utenti specifici è importante, ma ciò pone sfide per la privacy. L'apprendimento sul dispositivo consente la personalizzazione senza trasmettere così tanti dati privati. Tuttavia, bilanciare il miglioramento del modello, la tutela della privacy e i vincoli richiede nuove tecniche.

13.6.8 Considerazioni sulla Sicurezza

Se un ML embedded estremamente grande in sistemi come i veicoli a guida autonoma non viene progettato con attenzione, ci sono seri rischi per la sicurezza. Per garantire un funzionamento sicuro prima dell'implementazione, le auto a guida autonoma devono essere sottoposte a test approfonditi in pista in scenari simulati di pioggia, neve e ostacoli. Ciò richiede una convalida approfondita, dispositivi di sicurezza, simulatori e conformità agli standard prima dell'implementazione.

13.6.9 Diversi Target Hardware

Esiste una vasta gamma di processori embedded, tra cui ARM, x86, acceleratori AI specializzati, FPGA, ecc. Supportare questa eterogeneità rende difficile l'implementazione. Abbiamo bisogno di strategie come framework standardizzati, test approfonditi e messa a punto del modello per ogni piattaforma. Ad esempio, un modello di rilevamento degli oggetti necessita di implementazioni efficienti su dispositivi embedded come Raspberry Pi, Nvidia Jetson e Google Edge TPU.

13.6.10 Copertura dei Test

Testare rigorosamente i casi limite è difficile con risorse di simulazione embedded limitate, ma test esaustivi sono fondamentali in sistemi come le auto a guida autonoma. Testare esaustivamente un modello di pilota automatico richiede milioni di chilometri simulati, esponendolo a eventi rari come guasti dei sensori. Pertanto, strategie come la generazione di dati sintetici, la simulazione distribuita e l'ingegneria del caos aiutano a migliorare la copertura.

13.6.11 Rilevamento della Deriva del Concetto

Con dati di monitoraggio limitati da ogni dispositivo remoto, rilevare cambiamenti nei dati di input nel tempo è molto più difficile. La deriva può portare a degradazioni delle prestazioni del modello. Sono necessari metodi "leggeri" per identificare quando è necessario un riaddestramento. Un modello che prevede i carichi della rete elettrica mostra un calo delle prestazioni man mano che i pattern di utilizzo cambiano nel tempo. Con i soli dati locali sui dispositivi, questa tendenza è difficile da individuare.

13.7 MLOps Tradizionali e MLOps Embedded

Negli MLOps tradizionali, i modelli ML vengono in genere distribuiti in ambienti basati su cloud o server, con risorse abbondanti come potenza di calcolo e memoria. Questi ambienti facilitano il funzionamento regolare di modelli complessi che richiedono risorse di calcolo significative. Ad esempio, un modello di riconoscimento delle immagini basato su cloud potrebbe essere utilizzato da una piattaforma di social media per contrassegnare automaticamente le foto con etichette pertinenti. In questo caso, il modello può sfruttare le vaste risorse disponibili nel cloud per elaborare in modo efficiente grandi quantità di dati.
D'altro canto, i MLOps embedded comportano la distribuzione di modelli ML su sistemi embedded, sistemi di calcolo specializzati progettati per eseguire funzioni specifiche all'interno di sistemi più grandi. I sistemi embedded sono in genere caratterizzati dalle loro risorse di calcolo e potenza limitate. Ad esempio, un modello ML potrebbe essere "embedded" in un termostato intelligente per ottimizzare il riscaldamento e il raffreddamento in base alle preferenze e alle abitudini dell'utente. Il modello deve essere ottimizzato per funzionare in modo efficiente sull'hardware limitato del termostato senza comprometterne le prestazioni o la precisione.
La differenza fondamentale tra i MLOps tradizionali e quelli embedded risiede nei vincoli di risorse del sistema embedded. Mentre gli MLOps tradizionali possono sfruttare abbondanti risorse cloud o server, gli MLOps embedded devono fare i conti con le limitazioni hardware su cui viene distribuito il modello. Ciò richiede un'attenta ottimizzazione e messa a punto del modello per garantire che possa fornire informazioni accurate e preziose entro i vincoli del sistema embedded.
Inoltre, gli MLOps embedded devono considerare le sfide uniche poste dall'integrazione dei modelli ML con altri componenti del sistema embedded. Ad esempio, il modello deve essere compatibile con il software e l'hardware del sistema e deve essere in grado di interfacciarsi senza problemi con altri componenti, come sensori o attuatori. Ciò richiede una profonda comprensione sia dei sistemi ML che di quelli integrati e una stretta collaborazione tra data scientist, ingegneri e altre parti interessate.
Quindi, mentre gli MLOps tradizionali e gli MLOps embedded condividono l'obiettivo comune di distribuire e mantenere modelli ML in ambienti di produzione, le sfide uniche poste dai sistemi embedded richiedono un approccio specializzato. Gli MLOps embedded devono bilanciare attentamente la necessità di accuratezza e prestazioni del modello con i vincoli dell'hardware su cui viene distribuito il modello. Ciò richiede una profonda comprensione sia dei sistemi ML che di quelli embedded e una stretta collaborazione tra i vari stakeholder per garantire l'integrazione di successo dei modelli ML nei sistemi embedded.
Questa volta, raggrupperemo i sottoargomenti in categorie più ampie per semplificare la struttura del nostro processo di pensiero su MLOps. Questa struttura aiuterà a comprendere come i diversi aspetti di MLOps siano interconnessi e perché ciascuno sia importante per il funzionamento efficiente dei sistemi ML mentre discutiamo le sfide nel contesto dei sistemi embedded.

- Gestione del Ciclo di Vita del Modello
    - Gestione dei Dati: Gestione dell'ingestione dei dati, convalida e controllo delle versioni.
    - Addestramento dei Modelli: Tecniche e pratiche per un addestramento dei modelli efficace e scalabile.
    - Valutazione dei Modelli: Strategie per testare e convalidare le prestazioni dei modelli.
    - Distribuzione dei modelli: Approcci per la distribuzione dei modelli in ambienti di produzione.

- Integrazione di Sviluppo e Operazioni
  - Pipeline CI/CD: Integrazione dei modelli ML in pipeline di integrazione e distribuzione continue.
  - Gestione dell'infrastruttura: Impostazione e manutenzione dell'infrastruttura necessaria per il training e la distribuzione dei modelli.
  - Comunicazione e Collaborazione: Garanzia di una comunicazione e collaborazione fluide tra data scientist, ingegneri ML e team operativi.
- Eccellenza operativa
  - Monitoraggio: Tecniche per il monitoraggio delle prestazioni dei modelli, della deriva dei dati e dello stato operativo.
  - Governance: Implementazione di policy per la verificabilità, la conformità e le considerazioni etiche dei modelli.

13.7.1 Gestione del Ciclo di Vita del Modello

13.7.1.1 Gestione dei Dati

Nei tradizionali MLOps centralizzati, i dati vengono aggregati in grandi dataset e data lake, poi elaborati su server cloud o "on-prem" [in sede]. Tuttavia, MLOps embedded si basa su dati decentralizzati da sensori locali sui dispositivi. I dispositivi raccolgono batch più piccoli di dati incrementali, spesso rumorosi e non strutturati. Con vincoli di connettività, questi dati non possono sempre essere trasmessi istantaneamente al cloud e devono essere memorizzati nella cache in modo intelligente ed elaborati all'edge.
A causa della potenza di calcolo limitata sui dispositivi embedded, i dati si possono solo preelaborare e pulire in modo minimo prima della trasmissione. Il filtraggio e l'elaborazione anticipati avvengono nei gateway edge per ridurre i carichi di trasmissione. Mentre si sfrutta l'archiviazione cloud, altre elaborazioni e archiviazioni avvengono all'edge per tenere conto della connettività intermittente. I dispositivi identificano e trasmettono solo i sottoinsiemi di dati più critici al cloud.
Anche l'etichettatura richiede un accesso centralizzato ai dati, che richiede tecniche più automatizzate come l'apprendimento federato, in cui i dispositivi etichettano in modo collaborativo i dati dei peer. Con i dispositivi edge personali, la privacy dei dati e le normative sono preoccupazioni critiche. La raccolta, la trasmissione e l'archiviazione dei dati devono essere sicure e conformi.
Ad esempio, uno smartwatch può raccogliere il conteggio dei passi giornalieri, la frequenza cardiaca e le coordinate GPS. Questi dati vengono memorizzati nella cache locale e trasmessi a un gateway edge quando è disponibile il WiFi: il gateway elabora e filtra i dati prima di sincronizzare i sottoinsiemi rilevanti con la piattaforma cloud per riaddestrare i modelli.

13.7.1.2 Addestramento del Modello

Nei tradizionali MLOps centralizzati, i modelli vengono addestrati utilizzando dati abbondanti tramite deep learning su server GPU cloud ad alta potenza. Tuttavia, glii MLOps embedded necessitano di maggiore supporto in termini di complessità del modello, disponibilità dei dati e risorse di elaborazione per l'addestramento.
Il volume di dati aggregati è molto più basso, spesso richiedendo tecniche come l'apprendimento federato tra dispositivi per creare set di addestramento. La natura specializzata dei dati edge limita anche i set di dati pubblici per il pre-addestramento. Per questioni di privacy, i campioni di dati devono essere strettamente controllati e resi anonimi ove possibile.

Inoltre, i modelli devono utilizzare architetture semplificate ottimizzate per hardware edge a bassa potenza. Date le limitazioni di elaborazione, le GPU di fascia alta sono inaccessibili per un deep learning intensivo. L'addestramento sfrutta server edge e cluster a bassa potenza con approcci distribuiti per spartire il carico.
Il "transfer learning" emerge come una strategia cruciale per affrontare la scarsità di dati e l'irregolarità nell'apprendimento automatico, in particolare negli scenari di edge computing. Come illustrato in Figura 13.5, questo approccio prevede il pre-training di modelli su grandi set di dati pubblici e la loro successiva messa a punto su dati edge specifici del dominio. La figura raffigura una rete neurale in cui i layer iniziali (da W_{A1} a W_{A4}), responsabili dell'estrazione delle feature generali, sono congelati (indicati da una linea tratteggiata verde). Questi layer conservano la conoscenza delle attività precedenti, accelerando l'apprendimento e riducendo i requisiti di risorse. Gli ultimi livelli (da W_{A5} a W_{A7}), oltre la linea tratteggiata blu, sono messi a punto per l'attività specifica, concentrandosi sull'apprendimento delle feature specifiche dell'attività.

Figura 13.5: Trasferimento dell'apprendimento in MLOps. Fonte: HarvardX.

Questo metodo non solo mitiga la scarsità di dati, ma si adatta anche alla natura decentralizzata dei dati embedded. Inoltre, tecniche come l'apprendimento incrementale sul dispositivo possono personalizzare ulteriormente i modelli in base a casi d'uso specifici. La mancanza di dati ampiamente etichettati in molti domini motiva anche l'uso di tecniche semi-supervisionate, che completano l'approccio di apprendimento per trasferimento. Sfruttando le conoscenze preesistenti e adattandole a compiti specializzati, l'apprendimento per trasferimento all'interno di un framework MLOps consente ai modelli di ottenere prestazioni più elevate con meno risorse, anche in ambienti con vincoli di dati.
Ad esempio, un assistente domestico intelligente può pre-addestrare un modello di riconoscimento audio su clip YouTube pubbliche, il che aiuta a eseguire il bootstrap con conoscenze generali. Quindi trasferisce l'apprendimento a un piccolo campione di dati domestici per classificare elettrodomestici ed eventi personalizzati, specializzandosi nel modello. Il modello si trasforma in una rete neurale leggera ottimizzata per dispositivi abilitati al microfono in tutta la casa.
Pertanto, gli MLOps embedded affrontano sfide acute nella costruzione di set di dati di training, nella progettazione di modelli efficienti e nella distribuzione del calcolo per lo sviluppo del modello rispetto alle impostazioni tradizionali. Dati i vincoli embedded, è necessario un attento adattamento, come l'apprendimento tramite trasferimento e il training distribuito, per addestrare i modelli.

13.7.1.3 Valutazione del Modello

Nei tradizionali MLOps centralizzati, i modelli vengono valutati principalmente utilizzando metriche di accuratezza e dataset di test di holdout. Tuttavia, gli MLOps embedded richiedono una valutazione più olistica che tenga conto dei vincoli di sistema oltre all'accuratezza.

I modelli devono essere testati in anticipo e spesso su hardware edge distribuito che copre diverse configurazioni. Oltre all'accuratezza, fattori come latenza, utilizzo della CPU, ingombro di memoria e consumo energetico sono criteri di valutazione critici. I modelli vengono selezionati in base a compromessi tra queste metriche per soddisfare i vincoli dei dispositivi edge.

Anche la deriva dei dati deve essere monitorata, dove i modelli addestrati sui dati cloud degradano in accuratezza nel tempo sui dati edge locali. I dati embedded hanno spesso una maggiore variabilità rispetto ai set di addestramento centralizzati. Valutare i modelli su diversi campioni di dati edge operativi è fondamentale. Ma a volte, ottenere i dati per monitorare la deriva può essere difficile se questi dispositivi sono in circolazione e la comunicazione è una barriera.

Il monitoraggio continuo fornisce visibilità sulle prestazioni del mondo reale dopo l'implementazione, rivelando colli di bottiglia non evidenziati durante i test. Ad esempio, un aggiornamento del modello di una smart camera potrebbe essere inizialmente testato su 100 telecamere e poi annullato se si osserva un calo della precisione, prima di essere esteso a tutte le 5000 telecamere.

13.7.1.4 Distribuzione del Modello

Negli MLOps tradizionali, le nuove versioni del modello vengono distribuite direttamente sui server tramite endpoint API. Tuttavia, i dispositivi embedded richiedono meccanismi di distribuzione ottimizzati per ricevere modelli aggiornati. Gli aggiornamenti over-the-air (OTA) forniscono un approccio standardizzato alla distribuzione wireless di nuove versioni di software o firmware ai dispositivi embedded. Invece dell'accesso API diretto, i pacchetti OTA consentono la distribuzione remota di modelli e dipendenze come bundle pre-costruiti. In alternativa, l'apprendimento federato consente aggiornamenti del modello senza accesso diretto ai dati di training grezzi. Questo approccio decentralizzato ha il potenziale per un miglioramento continuo del modello, ma necessita di piattaforme MLOps robuste.

La distribuzione del modello si basa su interfacce fisiche come connessioni seriali USB o UART per dispositivi profondamente embedded privi di connettività. Il packaging del modello segue ancora principi simili agli aggiornamenti OTA, ma il meccanismo di distribuzione è adattato alle capacità dell'hardware edge. Inoltre, spesso vengono utilizzati protocolli OTA specializzati ottimizzati per reti IoT anziché protocolli WiFi o Bluetooth standard. I fattori chiave includono efficienza, affidabilità, sicurezza e telemetria, come il monitoraggio dei progressi, soluzioni come Mender. Io fornisce servizi OTA incentrati su embedded che gestiscono aggiornamenti differenziali tra flotte di dispositivi.

Figura 13.6 presenta una panoramica di "Model Lifecycle Management" in un contesto MLOps, illustrando il flusso dallo sviluppo (in alto a sinistra) alla distribuzione e al monitoraggio (in basso a destra). Il processo inizia con lo sviluppo ML, in cui il codice e le configurazioni sono "version-controlled". La gestione dei dati e dei modelli è fondamentale per il processo, coinvolgendo set di dati e repository di funzionalità. Training continuo, conversione del modello e registro del modello sono fasi chiave nell'operazionalizzazione della training. La distribuzione del modello include la fornitura del modello e la gestione dei log di fornitura. Sono in atto meccanismi di allarme per segnalare i problemi, che alimentano il monitoraggio continuo per garantire le prestazioni e l'affidabilità del modello nel tempo.

Questo approccio integrato garantisce che i modelli siano sviluppati e mantenuti in modo efficace durante tutto il loro ciclo di vita.

Figura 13.6: Gestione del ciclo di vita del modello. Fonte: HarvardX.

13.7.2 Integrazione di Sviluppo e Operazioni

13.7.2.1 Pipeline CI/CD

Nelle MLOps tradizionali, una solida infrastruttura CI/CD come Jenkins e Kubernetes consente l'automazione della pipeline per la distribuzione di modelli su larga scala. Tuttavia, le MLOps embedded necessitano di questa infrastruttura centralizzata e di flussi di lavoro CI/CD più personalizzati per i dispositivi edge.

La creazione di pipeline CI/CD deve tenere conto di un panorama frammentato di diverse versioni hardware, firmware e vincoli di connettività. Non esiste una piattaforma standard per orchestrare le pipeline e il supporto degli strumenti è più limitato.

I test devono coprire in anticipo questo ampio spettro di dispositivi embedded target, il che è difficile senza un accesso centralizzato. Le aziende devono investire molto nell'acquisizione e nella gestione dell'infrastruttura di test nell'ecosistema embedded eterogeneo.

Gli aggiornamenti over-the-air richiedono la configurazione di server specializzati per distribuire in modo sicuro i bundle di modelli ai dispositivi sul campo. Anche le procedure di rollout e rollback devono essere attentamente personalizzate per particolari famiglie di dispositivi.

Con gli strumenti CI/CD tradizionali meno applicabili, le MLOps embedded si affidano maggiormente a script personalizzati e integrazione. Le aziende adottano approcci diversi, dai framework open source alle soluzioni completamente interne. Una stretta integrazione tra sviluppatori, ingegneri edge e clienti finali stabilisce processi di rilascio affidabili.

Pertanto, gli MLOps embedded non possono sfruttare l'infrastruttura cloud centralizzata per CI/CD. Le aziende combinano pipeline personalizzate, infrastruttura di test e distribuzione OTA per distribuire modelli su sistemi edge frammentati e disconnessi.

13.7.2.2 Gestione dell'Infrastruttura

Nei tradizionali MLOps centralizzati, l'infrastruttura comporta l'approvvigionamento di server cloud, GPU e reti ad alta larghezza di banda per carichi di lavoro intensivi come l'addestramento di modelli e la fornitura di previsioni su larga scala. Tuttavia, gli MLOps embedded richiedono un'infrastruttura più eterogenea che si estende su dispositivi edge, gateway e cloud.

I dispositivi edge come i sensori catturano e preelaborano i dati localmente prima della trasmissione intermittente per evitare di sovraccaricare le reti: i gateway aggregano ed elaborano i dati dei dispositivi prima di inviare sottoinsiemi selezionati al cloud per l'addestramento e l'analisi. Il cloud fornisce gestione centralizzata ed elaborazione supplementare.

Questa infrastruttura necessita di una stretta integrazione e bilanciamento dei carichi di elaborazione e comunicazione. La larghezza di banda di rete è limitata, il che richiede un attento filtraggio e compressione dei dati. Le capacità di elaborazione edge sono modeste rispetto al cloud, imponendo vincoli di ottimizzazione.

La gestione di aggiornamenti OTA sicuri su grandi flotte di dispositivi presenta sfide all'edge. I rollout devono essere incrementali e pronti per il rollback per una rapida mitigazione. Dato l'ambiente decentralizzato, l'aggiornamento dell'infrastruttura edge richiede coordinamento.

Ad esempio, un impianto industriale può eseguire l'elaborazione di base del segnale sui sensori prima di inviare i dati a un gateway on-prem. Il gateway gestisce l'aggregazione dei dati, il monitoraggio dell'infrastruttura e gli aggiornamenti OTA. Solo i dati curati vengono trasmessi al cloud per analisi avanzate e riaddestramento del modello.

MLOps embedded richiede una gestione olistica dell'infrastruttura distribuita che abbraccia edge vincolato, gateway e cloud centralizzato. I carichi di lavoro sono bilanciati tra i livelli tenendo conto delle sfide di connettività, elaborazione e sicurezza.

13.7.2.3 Comunicazione e Collaborazione

Nelle MLOps tradizionali, la collaborazione tende a concentrarsi su data scientist, ingegneri ML e team DevOps. Tuttavia, le MLOps embedded richiedono un coordinamento interfunzionale più stretto tra ruoli aggiuntivi per affrontare i vincoli di sistema.

Gli ingegneri edge ottimizzano le architetture dei modelli per gli ambienti hardware target. Forniscono feedback ai data scientist durante lo sviluppo in modo che i modelli si adattino anticipatamente alle capacità dei dispositivi. Analogamente, i team di prodotto definiscono i requisiti operativi informati dai contesti degli utenti finali.

Con più stakeholder nell'ecosistema embedded, i canali di comunicazione devono facilitare la condivisione delle informazioni tra team centralizzati e remoti. Il monitoraggio dei problemi e la gestione dei progetti garantiscono l'allineamento.

Gli strumenti collaborativi ottimizzano i modelli per dispositivi specifici. I data scientist possono registrare i problemi replicati dai dispositivi sul campo in modo che i modelli siano specializzati in dati di nicchia. L'accesso remoto ai dispositivi facilita il debug e la raccolta dati.

Ad esempio, i data scientist possono collaborare con i team sul campo che gestiscono flotte di turbine eoliche per recuperare campioni di dati operativi. Questi dati vengono utilizzati per specializzare i modelli rilevando anomalie specifiche per quella classe di turbine. Gli aggiornamenti dei modelli vengono testati in simulazioni e rivisti dagli ingegneri prima dell'implementazione sul campo.

Gli MLOps embedded impongono un coordinamento continuo tra data scientist, ingegneri, clienti finali e altre parti interessate durante l'intero ciclo di vita del ML. Grazie a una stretta collaborazione, i modelli possono essere personalizzati e ottimizzati per i dispositivi edge mirati.

13.7.3 Eccellenza operativa

13.7.3.1 Monitoraggio

Il monitoraggio MLOps tradizionale si concentra sul monitoraggio centralizzato dell'accuratezza del modello, delle metriche delle prestazioni e della deriva dei dati. Tuttavia, MLOps embedded deve tenere conto del monitoraggio decentralizzato su diversi dispositivi e ambienti edge.

I dispositivi edge richiedono una raccolta dati ottimizzata per trasmettere metriche di monitoraggio chiave senza sovraccaricare le reti. Le metriche aiutano a valutare le prestazioni del modello, i pattern di dati, l'utilizzo delle risorse e altri comportamenti sui dispositivi remoti.

Con una connettività limitata, vengono eseguite più analisi all'edge prima di aggregare le informazioni centralmente. I gateway svolgono un ruolo chiave nel monitoraggio dello stato di salute della flotta e nel coordinamento degli aggiornamenti software. Gli indicatori confermati vengono infine propagati al cloud.

Un'ampia copertura dei dispositivi è impegnativa ma critica. Possono sorgere problemi specifici per determinati tipi di dispositivi, quindi il monitoraggio deve coprire l'intero spettro. Le distribuzioni "canary" aiutano a testare i processi di monitoraggio prima del ridimensionamento.

Il rilevamento delle anomalie identifica gli incidenti che richiedono il rollback dei modelli o la riqualificazione su nuovi dati. Tuttavia, l'interpretazione degli allarmi richiede la comprensione dei contesti dei dispositivi univoci in base all'input di ingegneri e clienti.

Ad esempio, una casa automobilistica può monitorare i veicoli autonomi per gli indicatori di degradazione del modello utilizzando la memorizzazione nella cache, l'aggregazione e i flussi in tempo reale. Gli ingegneri valutano quando le anomalie identificate garantiscono gli aggiornamenti OTA per migliorare i modelli in base a fattori come la posizione e l'età del veicolo.

Il monitoraggio MLOps embedded fornisce osservabilità nelle prestazioni del modello e del sistema in ambienti edge decentralizzati. Un'attenta raccolta, analisi e collaborazione dei dati fornisce informazioni significative per mantenere l'affidabilità.

13.7.3.2 Governance

Nelle MLOps tradizionali, la governance si concentra sulla spiegabilità del modello, la correttezza e la conformità per i sistemi centralizzati. Tuttavia, le MLOps embedded devono anche affrontare le sfide di governance a livello di dispositivo relative alla privacy dei dati, alla sicurezza e alla protezione.

Con i sensori che raccolgono dati personali e sensibili, la governance dei dati locali sui dispositivi è fondamentale. I controlli di accesso ai dati, l'anonimizzazione e la memorizzazione nella cache crittografata aiutano ad affrontare i rischi per la privacy e la conformità come HIPAA e GDPR. Gli aggiornamenti devono mantenere patch e impostazioni di sicurezza.

La governance della sicurezza considera gli impatti fisici del comportamento difettoso del dispositivo. I guasti potrebbero causare condizioni non sicure in veicoli, fabbriche e sistemi critici. Ridondanza, sistemi di sicurezza e sistemi di allarme aiutano a mitigare i rischi.

La governance tradizionale, come il monitoraggio dei bias e la spiegabilità del modello, rimane imperativa ma è più difficile da implementare per l'intelligenza artificiale embedded. Anche dare un'occhiata ai modelli black-box su dispositivi a basso consumo pone delle sfide. Ad esempio, un dispositivo medico può cancellare i dati personali sul dispositivo prima della trasmissione. I rigidi protocolli di governance dei dati approvano gli aggiornamenti del modello. La spiegabilità del modello è limitata, ma l'attenzione è rivolta al rilevamento di comportamenti anomali. I sistemi di backup prevengono i guasti.

La governance MLOps embedded deve comprendere privacy, sicurezza, protezione, trasparenza ed etica. Sono necessarie tecniche specializzate e collaborazione di squadra per aiutare a stabilire fiducia e responsabilità all'interno di ambienti decentralizzati.

13.7.4 Confronto

Tabella 13.2 evidenzia le somiglianze e le differenze tra MLOps Tradizionali e MLOps Embedded sulla base di tutto ciò che abbiamo imparato finora:

Tabella 13.2: Confronto tra le pratiche MLOps Tradizionali e quelle MLOps Embedded.

Area	MLOps Tradizionali	MLOps Embedded
Gestione dei Dati	Grandi set di dati, data lake, feature store	Acquisizione dati sul dispositivo, edge caching ed elaborazione
Sviluppo del Modello	Sfrutta il deep learning, reti neurali complesse, addestramento GPU	Vincoli sulla complessità del modello, necessità di ottimizzazione
Distribuzione	Cluster di server, distribuzione cloud, bassa latenza su larga scala	Distribuzione OTA su dispositivi, connettività intermittente
Monitoraggio	Dashboard, log, allarmi per le prestazioni del modello cloud	Monitoraggio sul dispositivo di previsioni, utilizzo delle risorse
Riqualificazione	Ri-addestramento dei modelli su nuovi dati	Apprendimento federato da dispositivi, ri-addestramento edge
Infrastruttura	Infrastruttura cloud dinamica	Infrastruttura edge/cloud eterogenea
Collaborazione	Monitoraggio degli esperimenti condivisi e registro dei modelli	Collaborazione per l'ottimizzazione specifica del dispositivo

Quindi, mentre Embedded MLOps condivide i principi fondamentali di MLOps, si trova ad affrontare vincoli unici nell'adattare flussi di lavoro e infrastrutture specificamente per dispositivi edge con risorse limitate.

13.7.5 MLOps Tradizionali

Google, Microsoft e Amazon offrono la loro versione di servizi ML gestiti. Questi includono servizi che gestiscono il training e la sperimentazione dei modelli, l'hosting e il ridimensionamento dei modelli e il monitoraggio. Queste offerte sono disponibili tramite un'API e SDK client, nonché tramite interfacce utente Web. Sebbene sia possibile creare le proprie soluzioni MLOps end-to-end utilizzando parti di ciascuno, i maggiori vantaggi in termini di facilità d'uso derivano dal rimanere all'interno di un singolo ecosistema di provider per sfruttare le integrazioni tra servizi.

Le sezioni seguenti presentano una rapida panoramica dei servizi che rientrano in ogni parte del ciclo di vita MLOps descritto sopra, fornendo esempi da diversi provider. È importante notare che lo spazio MLOps si sta evolvendo rapidamente; nuove aziende e prodotti stanno entrando in scena a un ritmo rapido. Gli esempi menzionati non intendono fungere da approvazione delle offerte di aziende specifiche, ma piuttosto illustrare i tipi di soluzioni disponibili sul mercato.

13.7.5.1 Gestione dei Dati

L'archiviazione dei dati e il versioning sono elementi essenziali per qualsiasi offerta commerciale e la maggior parte sfrutta le soluzioni di archiviazione generiche esistenti come S3. Altri utilizzano opzioni più specializzate come l'archiviazione basata su git (ad esempio: Hugging Face's Dataset Hub). Questa è un'area in cui i provider semplificano il supporto delle opzioni di archiviazione dati dei concorrenti, poiché non vogliono che ciò rappresenti una barriera per l'adozione del resto dei loro servizi MLOps. Ad esempio, la pipeline di addestramento di Vertex AI supporta senza problemi i set di dati archiviati in S3, Google Cloud Buckets o Dataset Hub di Hugging Face.

13.7.5.2 Addestramento del Modello

I servizi di training gestiti sono il punto di forza dei provider cloud, in quanto forniscono accesso on-demand a hardware che è fuori dalla portata della maggior parte delle aziende più piccole. Fatturano solo l'hardware per il tempo del training, accelerato con GPU accessibili anche ai team di sviluppatori più piccoli. Il controllo che gli sviluppatori hanno sul loro flusso di lavoro del training può variare notevolmente a seconda delle loro esigenze. Alcuni provider hanno servizi che forniscono poco più dell'accesso alle risorse e si affidano allo sviluppatore per gestire autonomamente il ciclo di training, il logging e l'archiviazione dei modelli. Altri servizi sono semplici come puntare a un modello di base e a un set di dati etichettato per avviare un lavoro di messa a punto completamente gestito (ad esempio: Vertex AI Fine Tuning).

Una parola di avvertimento: A partire dal 2023, la domanda di hardware GPU supera di gran lunga l'offerta e, di conseguenza, i provider cloud stanno razionando l'accesso alle loro GPU. In alcune regioni dei data center, le GPU potrebbero non essere disponibili o richiedere contratti a lungo termine.

13.7.5.3 Valutazione del Modello

Le attività di valutazione del modello in genere comportano il monitoraggio dell'accuratezza, della latenza e dell'utilizzo delle risorse dei modelli sia nelle fasi di test che di produzione. A differenza dei sistemi embedded, i modelli ML distribuiti sul cloud beneficiano di una connettività Internet costante e di capacità di logging illimitate. Di conseguenza, è spesso possibile acquisire e loggare ogni richiesta e risposta. Ciò rende trattabile la riproduzione o la generazione di richieste sintetiche per confrontare modelli e versioni diversi.

Alcuni provider offrono anche servizi che automatizzano il monitoraggio degli esperimenti di modifica degli iperparametri del modello. Tracciano le esecuzioni e le prestazioni e generano artefatti da queste esecuzioni di training del modello. Esempio: WeightsAndBiases

13.7.5.4 Distribuzione del Modello

Ogni provider in genere ha un servizio denominato "model registry", in cui vengono archiviati e a cui si accede ai modelli di training. Spesso, questi registri possono anche fornire accesso a modelli di base che sono open source o forniti da grandi aziende tecnologiche (o, in alcuni casi, come LLAMA, entrambi!). Questi registri dei modelli costituiscono un luogo comune per confrontare tutti i modelli e le loro versioni per consentire un facile processo decisionale su quale scegliere per un dato caso d'uso. Esempio: Vertex AI's model registry

Dal registro dei modelli, distribuire un modello a un endpoint di inferenza è rapido e semplice, e gestisce il provisioning delle risorse, il download del peso del modello e l'hosting di un dato modello. Questi servizi in genere forniscono accesso al modello tramite un'API REST con cui possono essere inviate richieste di inferenza. A seconda del tipo di modello, è possibile configurare risorse specifiche, ad esempio quale tipo di acceleratore GPU potrebbe essere necessario per raggiungere le prestazioni desiderate. Alcuni provider possono anche offrire opzioni di inferenza "serverless" o batch che non necessitano di un endpoint persistente per accedere al modello. Esempio: AWS SageMaker Inference

13.7.6 MLOps Embedded

Nonostante la proliferazione di nuovi strumenti MLOps in risposta all'aumento della domanda, le sfide descritte in precedenza hanno limitato la disponibilità di tali strumenti negli ambienti dei sistemi embedded. Più di recente, nuovi strumenti come Edge Impulse (Janapa Reddi et al. 2023) hanno reso il processo di sviluppo un po' più semplice, come descritto di seguito.

13.7.6.1 Edge Impulse

Edge Impulse è una piattaforma di sviluppo end-to-end per la creazione e l'implementazione di modelli di apprendimento automatico su dispositivi edge come microcontrollori e piccoli processori. Rende l'apprendimento automatico embedded più accessibile agli sviluppatori di software attraverso la sua interfaccia web di facile utilizzo e strumenti integrati per la raccolta dati, lo sviluppo di modelli, l'ottimizzazione e l'implementazione. Le sue funzionalità principali includono quanto segue:

- Flusso di lavoro intuitivo drag-and-drop per la creazione di modelli ML senza bisogno di codifica
- Strumenti per l'acquisizione, l'etichettatura, la visualizzazione e la preelaborazione dei dati dai sensori
- Scelta di architetture di modelli, tra cui reti neurali e apprendimento non supervisionato
- Tecniche di ottimizzazione dei modelli per bilanciare metriche delle prestazioni e vincoli hardware
- Distribuzione senza soluzione di continuità su dispositivi edge tramite compilazione, SDK e benchmark
- Funzionalità di collaborazione per team e integrazione con altre piattaforme

Edge Impulse offre una soluzione completa per la creazione di intelligenza embedded e l'avanzamento dell'apprendimento automatico, in particolare per gli sviluppatori con competenze limitate in scienza dei dati. Questa piattaforma consente lo sviluppo di modelli ML specializzati che funzionano in modo efficiente in piccoli ambienti di elaborazione. Come illustrato in Figura 13.7, Edge Impulse facilita il percorso dalla raccolta dati alla distribuzione del modello, evidenziando la sua interfaccia intuitiva e gli strumenti che semplificano la creazione di soluzioni ML incorporate, rendendole così accessibili a una gamma più ampia di sviluppatori e applicazioni.

Figura 13.7: Panoramica di Edge Impulse. Fonte: Edge Impulse

13.7.6.1.1 Interfaccia utente

Edge Impulse è stato progettato con sette principi chiave: accessibilità, funzionalità end-to-end, un approccio incentrato sui dati, interattività, estensibilità, orientamento al team e supporto della community. L'interfaccia utente intuitiva, mostrata in Figura 13.8, guida gli sviluppatori di tutti i livelli di esperienza attraverso il caricamento dei dati, la selezione di un'architettura di modello, l'addestramento del modello e la sua distribuzione su piattaforme hardware pertinenti. Va notato che, come qualsiasi strumento, Edge Impulse è destinato ad assistere, non a sostituire, le considerazioni fondamentali come la determinazione se ML è una soluzione appropriata o l'acquisizione delle competenze di dominio richieste per una determinata applicazione.

Figura 13.8: Schermata dell'interfaccia utente di Edge Impulse per la creazione di flussi di lavoro dai dati di input alle funzionalità di output.

Ciò che rende Edge Impulse degno di nota è il suo flusso di lavoro end-to-end completo ma intuitivo. Gli sviluppatori iniziano caricando i propri dati tramite l'interfaccia utente grafica (GUI) o gli strumenti dell'interfaccia a riga di comando (CLI), dopodiché possono esaminare campioni grezzi e visualizzare la distribuzione dei dati nelle suddivisioni di addestramento e test. Successivamente, gli utenti possono scegliere tra vari "blocchi" di pre-elaborazione per facilitare l'elaborazione del segnale digitale (DSP). Mentre vengono forniti valori di parametri predefiniti, gli utenti possono personalizzare i parametri in base alle proprie esigenze, osservando le considerazioni su memoria e la latenza visualizzate. Gli utenti possono scegliere facilmente la propria architettura di rete neurale, senza bisogno di alcun codice. Grazie all'editor visivo della piattaforma, gli utenti possono personalizzare i componenti dell'architettura e i parametri specifici, assicurandosi al contempo che il modello sia ancora addestrabile. Gli utenti possono anche sfruttare algoritmi di apprendimento non supervisionato, come il clustering K-means e i Gaussian Mixture Model (GMM).

13.7.6.1.2 Ottimizzazioni

Per adattarsi ai vincoli di risorse delle applicazioni TinyML, Edge Impulse fornisce una "matrice di confusione" che riassume le metriche chiave delle prestazioni, tra cui accuratezza per classe e punteggi F1. La piattaforma chiarisce i compromessi tra prestazioni del modello, dimensioni e latenza utilizzando simulazioni in Renode e benchmarking specifici del dispositivo. Per i casi di utilizzo dei dati in streaming, uno strumento di calibrazione delle prestazioni sfrutta un algoritmo genetico per trovare configurazioni di post-elaborazione ideali che bilanciano tassi di falsa accettazione e falso rifiuto. Sono disponibili tecniche come quantizzazione, ottimizzazione del codice e ottimizzazione specifica del dispositivo per i

modelli. Per la distribuzione, i modelli possono essere compilati in formati appropriati per i dispositivi edge target. Gli SDK del firmware nativi consentono anche la raccolta diretta dei dati sui dispositivi.
Oltre a semplificare lo sviluppo, Edge Impulse ridimensiona il processo di modellazione stesso. Una funzionalità chiave è EON Tuner, uno strumento di apprendimento automatico automatico (AutoML) che assiste gli utenti nell'ottimizzazione degli iperparametri in base ai vincoli di sistema. Esegue una ricerca casuale per generare rapidamente configurazioni per l'elaborazione del segnale digitale e le fasi di training. I modelli risultanti vengono visualizzati affinché l'utente possa selezionarli in base a metriche di prestazioni, memoria e latenza pertinenti. Per i dati, l'apprendimento attivo facilita il training su un piccolo sottoinsieme etichettato, seguito dall'etichettatura manuale o automatica di nuovi campioni in base alla vicinanza alle classi esistenti. Ciò espande l'efficienza dei dati.

13.7.6.1.3 Casi d'Uso

Oltre all'accessibilità della piattaforma stessa, il team di Edge Impulse ha ampliato la base di conoscenza dell'ecosistema ML embedded. La piattaforma si presta ad ambienti accademici, essendo stata utilizzata in corsi online e workshop in loco a livello globale. Sono stati pubblicati numerosi casi di studio con casi d'uso di settore e di ricerca, in particolare Oura Ring, che utilizza ML per identificare i pattern del sonno. Il team ha reso i repository open source su GitHub, facilitando la crescita della comunità. Gli utenti possono anche rendere pubblici i progetti per condividere tecniche e scaricare librerie da condividere tramite Apache. L'accesso a livello di organizzazione consente la collaborazione sui flussi di lavoro.
Nel complesso, Edge Impulse è straordinariamente completo e integrabile per i flussi di lavoro degli sviluppatori. Piattaforme più grandi come Google e Microsoft si concentrano maggiormente sul cloud rispetto ai sistemi embedded. I framework TinyMLOps come Neuton AI e Latent AI offrono alcune funzionalità ma non hanno le capacità end-to-end di Edge Impulse. TensorFlow Lite Micro è il motore di inferenza standard grazie alla flessibilità, allo stato open source e all'integrazione di TensorFlow, ma utilizza più memoria e storage rispetto al compilatore EON di Edge Impulse. Altre piattaforme devono essere aggiornate, focalizzate sull'aspetto accademico o più versatili. In sintesi, Edge Impulse semplifica e amplia l'apprendimento automatico embedded tramite una piattaforma accessibile e automatizzata.

13.7.6.2 Limitazioni

Sebbene Edge Impulse fornisca una pipeline accessibile per ML embedded, permangono importanti limitazioni e rischi. Una sfida fondamentale è la qualità e la disponibilità dei dati: i modelli sono validi solo quanto i dati utilizzati per addestrarli. Gli utenti devono disporre di campioni etichettati sufficienti che catturino l'ampiezza delle condizioni operative previste e delle modalità di errore. Le anomalie e i valori anomali etichettati sono critici, ma richiedono molto tempo per essere raccolti e identificati. Dati insufficienti o distorti comportano scarse prestazioni del modello indipendentemente dalle capacità dello strumento.
Anche il deploying su dispositivi a bassa potenza presenta sfide intrinseche. I modelli ottimizzati potrebbero comunque dover richiedere più risorse per MCU a bassissimo consumo. Trovare il giusto equilibrio tra compressione e accuratezza richiede un po' di sperimentazione. Lo strumento semplifica, ma deve comunque eliminare la necessità di competenze di base in ML ed elaborazione del segnale. Gli ambienti embedded limitano anche il debug e l'interpretabilità rispetto al cloud.
Sebbene siano ottenibili risultati impressionanti, gli utenti non dovrebbero considerare Edge Impulse come una soluzione "Push Button ML". Un'attenta definizione dell'ambito del progetto, la raccolta dati, la valutazione del modello e il test sono comunque essenziali. Come con qualsiasi strumento di sviluppo, si consigliano aspettative ragionevoli e diligenza

nell'applicazione. Tuttavia, Edge Impulse può accelerare la prototipazione e l'implementazione di ML embedded per gli sviluppatori disposti a investire lo sforzo di data science e ingegneria richiesto.

Esercizio 13.1: Edge Impulse

Pronti a far salire di livello i vostri piccoli progetti di machine-learning? Combiniamo la potenza di Edge Impulse con le fantastiche visualizzazioni di Weights & Biases (WandB). In questo Colab, si imparerà a monitorare i progressi del training del modello come un professionista! Si immagini di vedere fantastici grafici del modello che diventa più intelligente, confrontando diverse versioni e assicurandovi che la vostra IA funzioni al meglio anche su dispositivi minuscoli.

Open in Colab

13.8 Casi di Studio

13.8.1 Oura Ring

Oura Ring è un dispositivo indossabile che può misurare l'attività, il sonno e il recupero quando viene posizionato sul dito dell'utente. Utilizzando sensori per tracciare le metriche fisiologiche, il dispositivo utilizza ML embedded per prevedere le fasi del sonno. Per stabilire una base di legittimità nel settore, Oura ha condotto un esperimento di correlazione per valutare il successo del dispositivo nel prevedere le fasi del sonno rispetto a uno studio di base. Ciò ha portato a una solida correlazione del 62% rispetto alla base di riferimento dell'82-83%. Pertanto, il team ha deciso di determinare come migliorare ulteriormente le proprie prestazioni.

La prima sfida è stata ottenere dati migliori in termini sia di quantità che di qualità. Avrebbero potuto ospitare uno studio più ampio per ottenere un set di dati più completo, ma i dati sarebbero stati così rumorosi e grandi che sarebbe stato difficile aggregarli, ripulirli e analizzarli. È qui che entra in gioco Edge Impulse.

Abbiamo condotto un massiccio studio sul sonno su 100 uomini e donne di età compresa tra 15 e 73 anni in tre continenti (Asia, Europa e Nord America). Oltre a indossare l'Oura Ring, i partecipanti erano tenuti a sottoporsi al test PSG [https://it.wikipedia.org/wiki/Polisonnografia] standard del settore, che ha fornito una "etichetta" per questo set di dati. Con 440 notti di sonno da parte di 106 partecipanti, il set di dati ha totalizzato 3.444 ore di lunghezza tra dati Ring e PSG. Con Edge Impulse, Oura ha potuto caricare e consolidare facilmente i dati da diverse fonti in un bucket S3 privato. Sono stati anche in grado di impostare una Data Pipeline per unire campioni di dati in file individuali e preelaborare i dati senza dover eseguire lo "scrubbing" [pulizia] manuale.

Col tempo risparmiato nell'elaborazione dei dati grazie a Edge Impulse, il team Oura ha potuto concentrarsi sui driver chiave della propria previsione. Hanno estratto solo tre tipi di dati dei sensori: frequenza cardiaca, movimento e temperatura corporea. Dopo aver suddiviso i dati utilizzando la validazione incrociata a cinque livelli e classificato le fasi del sonno, il team ha ottenuto una correlazione del 79%, solo pochi punti percentuali in meno rispetto allo standard. Hanno prontamente distribuito due tipi di modelli di rilevamento del sonno: uno semplificato utilizzando solo l'accelerometro dell'anello e uno più completo sfruttando i segnali periferici mediati dal Autonomic Nervous System (ANS) [sistema nervoso autonomo] e le caratteristiche circadiane [ritmo cardiaco in 24 ore]. Con Edge Impulse, hanno in programma di condurre ulteriori analisi di diversi tipi di attività e sfruttare la scalabilità della piattaforma per continuare a sperimentare con diverse fonti di dati e sottoinsiemi di caratteristiche estratte.

Mentre la maggior parte della ricerca in ambito ML si concentra su fasi dominate dal modello, come il training e la messa a punto, questo caso di studio sottolinea l'importanza di un approccio olistico alle MLOps, in cui anche le fasi iniziali di aggregazione e pre-elaborazione dei dati hanno un impatto fondamentale sui risultati positivi.

13.8.2 ClinAIOps

Diamo un'occhiata a MLOps nel contesto del monitoraggio medico sanitario per comprendere meglio come MLOps "maturi" in un'implementazione nel mondo reale. In particolare, prendiamo in considerazione il continuous therapeutic monitoring (CTM) [monitoraggio terapeutico continuo] abilitato da dispositivi e sensori indossabili. Il CTM cattura dati fisiologici dettagliati dai pazienti, offrendo l'opportunità di aggiustamenti più frequenti e personalizzati ai trattamenti.

I sensori indossabili abilitati per ML consentono un monitoraggio continuo fisiologico e dell'attività al di fuori delle cliniche, aprendo possibilità per aggiustamenti terapeutici tempestivi e basati sui dati. Ad esempio, i biosensori indossabili per l'insulina (Psoma e Kanthou 2023) e i sensori ECG da polso per il monitoraggio del glucosio (J. Li et al. 2021) possono automatizzare il dosaggio di insulina per il diabete, i sensori ECG e PPG da polso possono regolare gli anticoagulanti in base ai pattern di fibrillazione atriale (Attia et al. 2018; Y. Guo et al. 2019), e gli accelerometri che tracciano l'andatura possono innescare cure preventive per la mobilità in declino negli anziani (Yingcheng Liu et al. 2022). La varietà di segnali che ora possono essere catturati passivamente e continuamente consente la titolazione e l'ottimizzazione della terapia su misura per le mutevoli esigenze di ogni paziente. Chiudendo il cerchio tra rilevamento fisiologico e risposta terapeutica con TinyML e apprendimento sul dispositivo, i dispositivi indossabili sono pronti a trasformare molte aree della medicina personalizzata.

Il ML è molto promettente nell'analisi dei dati CTM per fornire raccomandazioni basate sui dati per gli aggiustamenti della terapia. Ma semplicemente distribuire modelli di intelligenza artificiale in "silos", senza integrarli correttamente nei flussi di lavoro clinici e nel processo decisionale, può portare a una scarsa adozione o a risultati non ottimali. In altre parole, pensare solo a MLOps non è sufficiente per renderli utili nella pratica. Questo studio dimostra che sono necessari framework per incorporare intelligenza artificiale e CTM nella pratica clinica reale senza soluzione di continuità.

Questo caso di studio analizza "ClinAIOps" come modello per operazioni ML embedded in ambienti clinici complessi (E. Chen et al. 2023). Forniamo una panoramica del framework e del motivo per cui è necessario, esaminiamo un esempio di applicazione e discutiamo le principali sfide di implementazione relative al monitoraggio del modello, all'integrazione del flusso di lavoro e agli incentivi per gli stakeholder. L'analisi di esempi concreti come ClinAIOps illumina principi cruciali e best practice per operazioni AI affidabili ed efficaci in molti domini.

I framework MLOps tradizionali non sono sufficienti per integrare il monitoraggio terapeutico continuo (CTM) e l'IA in contesti clinici per alcuni motivi chiave:

- MLOps si concentra sul ciclo di vita del modello ML: training, distribuzione, monitoraggio. Ma l'assistenza sanitaria implica il coordinamento di più stakeholder umani, pazienti e medici, non solo modelli.
- MLOps automatizza il monitoraggio e la gestione dei sistemi IT. Tuttavia, l'ottimizzazione della salute del paziente richiede cure personalizzate e supervisione umana, non solo automazione.
- CTM e l'erogazione dell'assistenza sanitaria sono sistemi sociotecnici complessi con molte parti mobili. MLOps non fornisce un framework per coordinare il processo decisionale umano e AI.

- Le considerazioni etiche relative all'AI sanitaria richiedono giudizio umano, supervisione e responsabilità. I framework MLOps non hanno processi per la supervisione etica.
- I dati sanitari dei pazienti sono altamente sensibili e regolamentati. MLOps da solo non garantisce la gestione delle informazioni sanitarie protette secondo gli standard normativi e di privacy.
- La convalida clinica dei piani di trattamento guidati dall'AI è essenziale per l'adozione da parte del provider. MLOps non incorpora la valutazione specifica del dominio delle raccomandazioni del modello.
- L'ottimizzazione delle metriche sanitarie come i risultati dei pazienti richiede l'allineamento degli incentivi e dei flussi di lavoro delle parti interessate, che MLOps puramente incentrato sulla tecnologia trascura.

Pertanto, integrare efficacemente AI/ML e CTM nella pratica clinica richiede più di semplici modelli e pipeline di dati; richiede il coordinamento di complessi processi decisionali collaborativi tra esseri umani e AI, che ClinAIOps affronta tramite i suoi cicli di feedback multi-stakeholder.

13.8.2.1 Cicli di Feedback

Il framework ClinAIOps, mostrato in Figura 13.9, fornisce questi meccanismi attraverso tre cicli di feedback. I cicli sono utili per coordinare le informazioni dal monitoraggio fisiologico continuo, l'esperienza del medico e la guida dell'IA tramite cicli di feedback, consentendo una medicina di precisione basata sui dati mantenendo al contempo la responsabilità umana. ClinAIOps fornisce un modello per un'efficace simbiosi uomo-IA nell'assistenza sanitaria: il paziente è al centro, fornendo sfide e obiettivi sanitari che informano il regime terapeutico; il medico supervisiona questo regime, fornendo input per gli aggiustamenti basati sui dati di monitoraggio continuo e sui report sanitari del paziente; mentre gli sviluppatori di IA svolgono un ruolo cruciale creando sistemi che generano allarmi per gli aggiornamenti della terapia, che il medico quindi esamina.

Questi cicli di feedback, di cui parleremo di seguito, aiutano a mantenere la responsabilità e il controllo del medico sui piani di trattamento esaminando i suggerimenti dell'IA prima che abbiano un impatto sui pazienti. Aiutano a personalizzare dinamicamente il comportamento e gli output del modello di IA in base allo stato di salute mutevole di ciascun paziente. Contribuiscono a migliorare l'accuratezza del modello e l'utilità clinica nel tempo, imparando dalle risposte del medico e del paziente. Facilitano il processo decisionale condiviso e l'assistenza personalizzata durante le interazioni paziente-medico. Consentono una rapida ottimizzazione delle terapie in base a dati frequenti del paziente che i medici non possono analizzare manualmente.

Figura 13.9: Ciclo ClinAIOps. Fonte: E. Chen et al. (2023).

13.8.2.1.1 Ciclo Paziente-IA

Il ciclo paziente-IA consente un'ottimizzazione frequente della terapia guidata dal monitoraggio fisiologico continuo. Ai pazienti vengono prescritti dispositivi indossabili come smartwatch o cerotti cutanei per raccogliere passivamente segnali sanitari rilevanti. Ad esempio, un paziente diabetico potrebbe avere un monitoraggio continuo del glucosio o un paziente con malattie cardiache potrebbe indossare un cerotto ECG. Un modello di IA analizza i flussi di dati sanitari longitudinali del paziente nel contesto delle sue cartelle cliniche elettroniche: diagnosi, esami di laboratorio, farmaci e dati demografici. Il modello di IA suggerisce modifiche al regime di trattamento su misura per quell'individuo, come la modifica di una dose di farmaco o di un programma di somministrazione. Piccole modifiche entro un intervallo di sicurezza pre-approvato possono essere apportate dal paziente in modo indipendente, mentre le modifiche più importanti vengono prima esaminate dal medico. Questo stretto feedback tra la fisiologia del paziente e la terapia guidata dall'IA consente ottimizzazioni tempestive basate sui dati come raccomandazioni automatizzate sul dosaggio di insulina basate sui livelli di glucosio in tempo reale per i pazienti diabetici.

13.8.2.1.2 Ciclo Clinico-IA

Il ciclo clinico-IA consente la supervisione clinica sulle raccomandazioni generate dall'IA per garantire sicurezza e responsabilità. Il modello di IA fornisce al medico raccomandazioni terapeutiche e riepiloghi facilmente esaminabili dei dati rilevanti del paziente su cui si basano i suggerimenti. Ad esempio, un'IA può suggerire di ridurre la dose di farmaci per la pressione sanguigna di un paziente iperteso in base a letture costantemente basse. Il medico può accettare, rifiutare o modificare le modifiche alla prescrizione proposte dall'IA. Questo feedback del medico addestra e migliora ulteriormente il modello. Inoltre, il medico stabilisce i limiti per i tipi e l'entità delle modifiche al trattamento che l'IA può raccomandare autonomamente ai pazienti. Esaminando i suggerimenti dell'IA, il medico mantiene l'autorità di trattamento finale in base al proprio giudizio clinico e alla propria responsabilità. Questo ciclo consente loro di supervisionare i casi dei pazienti con l'assistenza dell'IA in modo efficiente.

13.8.2.1.3 Ciclo Paziente-Clinico

Invece di una raccolta dati di routine, il medico può concentrarsi sull'interpretazione di pattern di dati di alto livello e sulla collaborazione con il paziente per stabilire obiettivi e priorità di salute. L'assistenza AI libererà anche il tempo dei medici, consentendo loro di concentrarsi maggiormente sull'ascolto delle storie e delle preoccupazioni dei pazienti. Ad esempio, il medico può discutere di cambiamenti di dieta ed esercizio fisico con un paziente diabetico per migliorare il controllo del glucosio in base ai dati di monitoraggio continuo. La frequenza degli appuntamenti può anche essere regolata dinamicamente in base ai progressi del paziente anziché seguire un calendario fisso. Liberato dalla raccolta di dati di base, il medico può fornire "coaching" e cure personalizzate a ciascun paziente informato dai suoi dati sanitari continui. La relazione paziente-medico diventa più produttiva e personalizzata.

13.8.2.2 Esempio di Ipertensione

Consideriamo un esempio. Secondo i "Centers for Disease Control and Prevention", quasi la metà degli adulti soffre di ipertensione (48.1%, 119.9 milioni). L'ipertensione può essere gestita tramite ClinAIOps con l'aiuto di sensori indossabili utilizzando il seguente approccio:

13.8.2.2.1 Raccolta Dati

I dati raccolti includerebbero il monitoraggio continuo della pressione sanguigna tramite un dispositivo indossato al polso dotato di sensori per fotopletismografia (PPG) ed elettrocardiografia (ECG) per stimare la pressione sanguigna (Q. Zhang, Zhou, e Zeng 2017). Il dispositivo indossabile monitorerebbe anche l'attività fisica del paziente tramite accelerometri embedded. Il paziente registrerebbe tutti i farmaci antipertensivi assunti, insieme all'ora e alla dose. Verrebbero inoltre incorporati i dettagli demografici e la storia clinica del paziente dalla sua cartella clinica elettronica (EHR). Questi dati multimodali del mondo reale forniscono un contesto prezioso al modello di intelligenza artificiale per analizzare i pattern di pressione sanguigna del paziente, i livelli di attività, l'aderenza ai farmaci e le risposte alla terapia.

13.8.2.2.2 Modello di Intelligenza Artificiale

Il modello di intelligenza artificiale sul dispositivo analizzerebbe le tendenze continue della pressione sanguigna del paziente, i pattern circadiani, i livelli di attività fisica, i comportamenti di aderenza ai farmaci e altri contesti. Utilizzerebbe ML per prevedere dosi ottimali di farmaci antipertensivi e tempi per controllare la pressione sanguigna dell'individuo. Il modello invierebbe raccomandazioni di modifica del dosaggio direttamente al paziente per piccoli aggiustamenti o al medico revisore per l'approvazione per modifiche più significative. Osservando il feedback clinico sulle sue raccomandazioni e valutando i risultati ottenuti sulla pressione sanguigna nei pazienti, il modello di intelligenza artificiale potrebbe essere continuamente riqualificato per migliorarne le prestazioni. L'obiettivo è una gestione della pressione sanguigna completamente personalizzata ottimizzata per le esigenze e le risposte di ciascun paziente.

13.8.2.2.3 Ciclo Paziente-IA

Nel ciclo Paziente-IA, il paziente iperteso riceverebbe notifiche sul suo dispositivo indossabile o sull'app per smartphone collegata che raccomandano modifiche ai suoi farmaci antipertensivi. Per piccole modifiche della dose entro un intervallo di sicurezza predefinito, il paziente potrebbe implementare in modo indipendente la modifica suggerita dal modello di IA al suo regime. Tuttavia, il paziente deve ottenere l'approvazione del medico prima di modificare il dosaggio per modifiche più significative. Fornire raccomandazioni personalizzate e tempestive sui farmaci automatizza un elemento di autogestione dell'ipertensione per il paziente. Può migliorare la sua aderenza al regime e i risultati del

trattamento. Il paziente è autorizzato a sfruttare le informazioni dell'IA per controllare meglio la sua pressione sanguigna.

13.8.2.2.4 Ciclo Clinico-IA

Nel ciclo Clinico-IA, il fornitore riceverebbe riepiloghi delle tendenze continue della pressione sanguigna del paziente e visualizzazioni dei suoi pattern di assunzione dei farmaci e dell'aderenza. Esaminano le modifiche al dosaggio antipertensivo suggerite dal modello AI e decidono se approvare, rifiutare o modificare le raccomandazioni prima che raggiungano il paziente. Il medico specifica anche i limiti di quanto l'AI può raccomandare in modo indipendente di modificare i dosaggi senza la supervisione del medico. Se la pressione sanguigna del paziente tende a livelli pericolosi, il sistema avvisa il medico in modo che possa intervenire tempestivamente e modificare i farmaci o richiedere una visita al pronto soccorso. Questo ciclo mantiene responsabilità e sicurezza consentendo al contempo al medico di sfruttare le intuizioni dell'AI mantenendo il medico responsabile dell'approvazione delle principali modifiche al trattamento.

13.8.2.2.5 Ciclo Paziente-Clinico

Nel ciclo Paziente-Clinico, mostrato in Figura 13.10, le visite di persona si concentrerebbero meno sulla raccolta di dati o sulle modifiche di base dei farmaci. Invece, il medico potrebbe interpretare tendenze e pattern di alto livello nei dati di monitoraggio continuo del paziente e avere discussioni mirate su dieta, esercizio fisico, gestione dello stress e altri cambiamenti nello stile di vita per migliorare il controllo della pressione sanguigna in modo olistico. La frequenza degli appuntamenti potrebbe essere ottimizzata dinamicamente in base alla stabilità del paziente anziché seguire un calendario fisso. Poiché il medico non avrebbe bisogno di rivedere tutti i dati granulari, potrebbe concentrarsi sulla fornitura di cure e raccomandazioni personalizzate durante le visite. Con il monitoraggio continuo e l'ottimizzazione assistita dall'intelligenza artificiale dei farmaci tra le visite, la relazione medico-paziente si concentra sugli obiettivi di benessere generale e diventa più incisiva. Questo approccio proattivo e personalizzato basato sui dati può aiutare a evitare complicazioni dell'ipertensione come ictus, insufficienza cardiaca e altre minacce alla salute e al benessere del paziente.

Figura 13.10: Ciclo interattivo ClinAIOps. Fonte: E. Chen et al. (2023).

13.8.2.3 MLOps vs. ClinAIOps

L'esempio dell'ipertensione illustra bene perché i tradizionali MLOps sono insufficienti per molte applicazioni AI del mondo reale e perché sono invece necessari framework come ClinAIOps.

Con l'ipertensione, il semplice sviluppo e distribuzione di un modello ML per la regolazione dei farmaci avrebbe successo solo se considerasse il contesto clinico più ampio. Il paziente, il medico e il sistema sanitario hanno preoccupazioni sulla definizione dell'adozione. Il modello AI non può ottimizzare da solo i risultati della pressione sanguigna: richiede l'integrazione con flussi di lavoro, comportamenti e incentivi.

- Alcune lacune chiave evidenziate dall'esempio in un approccio MLOps puro:

- Il modello stesso non avrebbe i dati dei pazienti del mondo reale su larga scala per raccomandare trattamenti in modo affidabile. ClinAIOps consente ciò raccogliendo feedback da medici e pazienti tramite monitoraggio continuo.
- I medici si fiderebbero delle raccomandazioni del modello solo con trasparenza, spiegabilità e responsabilità. ClinAIOps mantiene il medico informato per creare fiducia.
- I pazienti hanno bisogno di coaching e motivazione personalizzati, non solo di notifiche AI. Il ciclo paziente-clinico di ClinAIOps facilita questo.
- L'affidabilità dei sensori e l'accuratezza dei dati sarebbero sufficienti solo con la supervisione clinica. ClinAIOps convalida le raccomandazioni.
- La responsabilità per i risultati del trattamento deve essere chiarita solo con un modello ML. ClinAIOps mantiene la responsabilità umana.
- I sistemi sanitari dovrebbero dimostrare il valore per cambiare i flussi di lavoro. ClinAIOps allinea le parti interessate.

Il caso dell'ipertensione mostra chiaramente la necessità di guardare oltre il training e l'implementazione di un modello ML performante per considerare l'intero sistema sociotecnico umano-IA. Questa è la lacuna principale che ClinAIOps colma rispetto ai tradizionali MLOps. I tradizionali MLOps sono eccessivamente focalizzati sulla tecnologia per automatizzare lo sviluppo e l'implementazione del modello ML, mentre ClinAIOps incorpora il contesto clinico e il coordinamento umano-IA attraverso cicli di feedback multi-stakeholder. Tabella 13.3 li confronta. Questa tabella evidenzia come, quando si implementa MLOps, sia necessario considerare più dei semplici modelli ML.

Tabella 13.3: Confronto tra operazioni MLOps e AI per uso clinico.

	MLOps tradizionali	**ClinAIOps**
Focus	Sviluppo e distribuzione di modelli ML	Coordinamento del processo decisionale umano e AI
Parti interessate	Data scientist, ingegneri IT	Pazienti, medici, sviluppatori AI
Cicli di feedback	Riqualificazione del modello, monitoraggio	Paziente-IA, clinico-IA, paziente-clinico
Obiettivo	Rendere operative le distribuzioni ML	Ottimizzare i risultati di salute del paziente
Processi	Pipeline e infrastruttura automatizzate	Integra flussi di lavoro clinici e supervisione
Considerazioni sui dati	Creazione di set di dati di training	Privacy, etica, informazioni sanitarie protette
Validazione del modello	Test delle metriche delle prestazioni del modello	Valutazione clinica delle raccomandazioni
Implementazione	Si concentra sull'integrazione tecnica	Allinea gli incentivi degli stakeholder umani

13.8.2.4 Riepilogo

In ambiti complessi come l'assistenza sanitaria, l'implementazione di successo dell'IA richiede di andare oltre un focus ristretto sul training e il deploying di modelli ML performanti. Come illustrato nell'esempio dell'ipertensione, l'integrazione dell'IA nel mondo reale richiede il coordinamento di diverse parti interessate, l'allineamento degli incentivi, la convalida delle raccomandazioni e il mantenimento della responsabilità. Framework come ClinAIOps, che facilitano il processo decisionale collaborativo tra uomo e IA attraverso cicli di feedback integrati, sono necessari per affrontare queste sfide multiformi. Invece di automatizzare semplicemente le attività, l'IA deve aumentare le capacità umane e i flussi di lavoro clinici.

Ciò consente all'IA di avere un impatto positivo sui risultati dei pazienti, sulla salute della popolazione e sull'efficienza dell'assistenza sanitaria.

13.9 Conclusione

L'ML embedded è pronto a trasformare molti settori abilitando le funzionalità AI direttamente su dispositivi edge come smartphone, sensori e hardware IoT. Tuttavia, lo sviluppo e l'implementazione di modelli TinyML su sistemi embedded con risorse limitate pone sfide uniche rispetto ai tradizionali MLOps basati su cloud.

Questo capitolo ha fornito un'analisi approfondita delle principali differenze tra MLOps tradizionali ed embedded nel ciclo di vita del modello, flussi di lavoro di sviluppo, gestione dell'infrastruttura e pratiche operative. Abbiamo discusso di come fattori come connettività intermittente, dati decentralizzati e computing limitato sul dispositivo richiedano tecniche innovative come apprendimento federato, inferenza sul dispositivo e ottimizzazione del modello. Pattern architettonici come apprendimento cross-device e infrastruttura edge-cloud gerarchica aiutano a mitigare i vincoli.

Attraverso esempi concreti come Oura Ring e ClinAIOps, abbiamo dimostrato i principi applicati per MLOps embedded. I casi di studio hanno evidenziato considerazioni critiche che vanno oltre l'ingegneria ML di base, come l'allineamento degli incentivi delle parti interessate, il mantenimento della responsabilità e il coordinamento del processo decisionale tra uomo e IA. Ciò sottolinea la necessità di un approccio olistico che abbracci sia gli elementi tecnici che quelli umani.

Mentre gli MLOps embedded incontrano degli ostacoli, strumenti emergenti come Edge Impulse e lezioni dai pionieri aiutano ad accelerare l'innovazione del TinyML. Una solida comprensione dei principi fondamentali degli MLOps adattati agli ambienti embedded consentirà a più organizzazioni di superare i vincoli e fornire capacità di intelligenza artificiale distribuita. Man mano che i framework e le best practice maturano, l'integrazione fluida dell'ML nei dispositivi e nei processi edge trasformerà i settori attraverso l'intelligenza localizzata.

13.10 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale al proprio ritmo. Incoraggiamo sia gli studenti che gli insegnanti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- MLOps, DevOps, and AIOps.
- MLOps overview.
- Tiny MLOps.
- MLOps: a use case.
- MLOps: Key Activities and Lifecycle.
- ML Lifecycle.
- Scaling TinyML: Challenges and Opportunities.
- Operazionalizzazione del Training:
  - Training Ops: CI/CD trigger.
  - Continuous Integration.
  - Continuous Deployment.
  - Production Deployment.
  - Production Deployment: Online Experimentation.
  - Training Ops Impact on MLOps.
- Deployment del Modello:
  - Scaling ML Into Production Deployment.
  - Containers for Scaling ML Deployment.
  - Challenges for Scaling TinyML Deployment: Part 1.
  - Challenges for Scaling TinyML Deployment: Part 2.
  - Model Deployment Impact on MLOps.

Video

- Video 13.1
- Video 13.2
- Video 13.3
- Video 13.4

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 13.1

⚠ Laboratori

Oltre agli esercizi, offriamo anche una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

14. Sicurezza e Privacy

DALL·E 3 Prompt: Un'illustrazione sulla privacy e la sicurezza nei sistemi di apprendimento automatico. L'immagine mostra un paesaggio digitale con una rete di nodi interconnessi e flussi di dati, che simboleggiano gli algoritmi di apprendimento automatico. In primo piano, c'è un grande lucchetto sovrapposto alla rete, che rappresenta la privacy e la sicurezza. Il lucchetto è semi-trasparente, consentendo alla rete sottostante di essere parzialmente visibile. Lo sfondo presenta codice binario e simboli di crittografia digitale, che enfatizzano il tema della sicurezza informatica. La combinazione di colori è un mix di blu, verdi e grigi, che suggerisce un ambiente digitale ad alta tecnologia.

Sicurezza e privacy sono fondamentali quando si sviluppano sistemi di apprendimento automatico nel mondo reale. Poiché l'apprendimento automatico viene sempre più applicato a domini sensibili come sanità, finanza e dati personali, proteggere la riservatezza e prevenire l'uso improprio di dati e modelli diventa imperativo. Chiunque intenda creare sistemi di apprendimento automatico solidi e responsabili deve comprendere i potenziali rischi per la sicurezza e la privacy, come perdite di dati, furto di modelli, attacchi avversari, bias [pregiudizi] e accesso involontario a informazioni private. Dobbiamo anche comprendere le best practice per mitigare questi rischi. Ancora più importante, sicurezza e privacy non possono essere un ripensamento e devono essere affrontate in modo proattivo durante tutto il ciclo di vita dello sviluppo del sistema di apprendimento automatico, dalla raccolta e dall'etichettatura dei dati al training, valutazione e deployment distribuzione del modello. Incorporare considerazioni sulla sicurezza e sulla privacy in ogni fase di creazione, distribuzione e gestione dei sistemi di apprendimento automatico è essenziale per sbloccare in modo sicuro i vantaggi dell'intelligenza artificiale.

Obiettivi dell'Apprendimento

- Comprendere i principali rischi per la privacy e la sicurezza del ML, come perdite di dati, furto di modelli, attacchi avversari, pregiudizi e accesso involontario ai dati.
- Imparare dagli incidenti storici di sicurezza di sistemi hardware ed embedded.
- Identificare le minacce ai modelli ML come avvelenamento dei dati, estrazione di modelli, inferenza di appartenenza ed esempi avversari.
- Riconoscere le minacce alla sicurezza hardware per il ML embedded che abbracciano bug hardware, attacchi fisici, canali laterali, componenti contraffatti, ecc.
- Esplorare le difese ML embedded, come ambienti di esecuzione affidabili, avvio sicuro, funzioni fisiche non clonabili e moduli di sicurezza hardware.
- Discutere i problemi di privacy nella gestione di dati utente sensibili con ML embedded, comprese le normative.
- Apprendere tecniche ML che preservano la privacy come la privacy differenziale, apprendimento federato, crittografia omomorfica e generazione di dati sintetici.
- Comprendere i compromessi tra privacy, accuratezza, efficienza, modelli di minaccia e ipotesi di fiducia.
- Riconoscere la necessità di una prospettiva multilivello che abbracci progettazione elettrica, firmware, software e fisica quando si proteggono dispositivi ML embedded.

14.1 Introduzione

Il "Machine learning" [apprendimento automatico] si è evoluto notevolmente dalle sue origini accademiche, in cui la privacy non era una preoccupazione primaria. Con la migrazione del ML in applicazioni commerciali e consumer, i dati sono diventati più sensibili, comprendendo informazioni personali come comunicazioni, acquisti e dati sanitari. Questa esplosione di disponibilità di dati ha alimentato rapidi progressi nelle capacità del ML. Tuttavia, ha anche esposto nuovi rischi per la privacy, come dimostrato da incidenti come la fuga di dati di AOL nel 2006 e lo scandalo Cambridge Analytica.

Questi eventi hanno evidenziato la crescente necessità di affrontare la privacy nei sistemi ML. In questo capitolo, esploriamo insieme considerazioni sulla privacy e sulla sicurezza, poiché sono intrinsecamente collegate nel ML:

- La privacy si riferisce al controllo dell'accesso ai dati sensibili degli utenti, come informazioni finanziarie o dati biometrici raccolti da un'applicazione ML.
- La sicurezza protegge i sistemi e i dati ML da hacking, furto e uso improprio.

Ad esempio, una telecamera di sicurezza domestica basata su ML deve proteggere i flussi video da accessi non autorizzati e fornire protezioni della privacy per garantire che solo gli utenti previsti possano visualizzare il filmato. Una violazione della sicurezza o della privacy potrebbe esporre momenti privati degli utenti.

I sistemi ML embedded come assistenti intelligenti e dispositivi indossabili sono onnipresenti ed elaborano dati intimi degli utenti. Tuttavia, i loro vincoli computazionali spesso impediscono protocolli di sicurezza pesanti. I progettisti devono bilanciare le esigenze di prestazioni con rigorosi standard di sicurezza e privacy adattati alle limitazioni dell'hardware embedded.

Questo capitolo fornisce conoscenze essenziali per affrontare il complesso panorama di privacy e sicurezza dell'ML embedded. Esploreremo le vulnerabilità e tratteremo varie tecniche che migliorano la privacy e la sicurezza all'interno dei vincoli di risorse dei sistemi embedded.

Ci auguriamo che sviluppando una comprensione olistica dei rischi e delle misure di sicurezza, si acquisiranno i principi per sviluppare applicazioni ML embedded sicure ed etiche.

14.2 Terminologia

In questo capitolo parleremo insieme di sicurezza e privacy, quindi ci sono termini chiave su cui dobbiamo essere chiari.

- **Privacy:** Si consideri una telecamera di sicurezza domestica basata su ML che identifica e registra potenziali minacce. Questa telecamera registra informazioni identificabili di individui che si avvicinano e potenzialmente entrano in questa casa, compresi i volti. Le preoccupazioni sulla privacy potrebbero riguardare chi può accedere a questi dati.
- **Sicurezza:** Si consideri una telecamera di sicurezza domestica basata su ML che identifica e registra potenziali minacce. L'aspetto della sicurezza garantirebbe che gli hacker non possano accedere a questi video e ai modelli di riconoscimento.
- **Minaccia:** Utilizzando il nostro esempio della telecamera di sicurezza domestica, una minaccia potrebbe essere un hacker che tenta di accedere a video live o archiviati o che utilizza falsi input per ingannare il sistema.
- **Vulnerabilità:** Una vulnerabilità comune potrebbe essere una rete scarsamente protetta tramite la quale la telecamera si connette a Internet, che potrebbe essere sfruttata per accedere ai dati.

14.3 Precedenti Storici

Sebbene le specifiche della sicurezza hardware dell'apprendimento automatico possano essere distinte, il campo dei sistemi embedded ha una storia di incidenti di sicurezza che forniscono lezioni fondamentali per tutti i sistemi connessi, compresi quelli che utilizzano ML. Ecco analisi dettagliate di violazioni passate:

14.3.1 Stuxnet

Nel 2010, qualcosa di inaspettato è stato trovato su un computer in Iran: un virus informatico molto complicato che gli esperti non avevano mai visto prima. Stuxnet era un worm informatico dannoso che prendeva di mira i sistemi di controllo di supervisione e acquisizione dati (SCADA) ed era progettato per danneggiare il programma nucleare iraniano (Farwell e Rohozinski 2011). Stuxnet stava utilizzando quattro "exploit zero-day", attacchi che sfruttano debolezze segrete nel software di cui nessuno è ancora a conoscenza. Ciò ha reso Stuxnet molto subdolo e difficile da rilevare.

Ma Stuxnet non è stato progettato per rubare informazioni o spiare le persone. Il suo obiettivo era la distruzione fisica, sabotare le centrifughe della centrale nucleare iraniana di Natanz! Quindi come ha fatto il virus a raggiungere i computer della centrale di Natanz, che avrebbe dovuto essere disconnessa dal mondo esterno per motivi di sicurezza? Gli esperti pensano che qualcuno abbia inserito una chiavetta USB contenente Stuxnet nella rete interna di Natanz. Ciò ha permesso al virus di "saltare" da un sistema esterno ai sistemi di controllo nucleare isolati e scatenare il caos.

Stuxnet era un malware incredibilmente avanzato creato dai governi nazionali per passare dal regno digitale alle infrastrutture del mondo reale. Ha preso di mira in modo specifico importanti macchine industriali, dove l'apprendimento automatico embedded è altamente applicabile in un modo mai visto prima. Il virus ha lanciato un segnale di allarme su come i sofisticati attacchi informatici potrebbero ora distruggere fisicamente apparecchiature e strutture.

Questa violazione è stata significativa a causa della sua sofisticatezza; Stuxnet ha preso di mira in modo specifico i "programmable logic controllers (PLC)" utilizzati per automatizzare processi elettromeccanici come la velocità delle centrifughe per l'arricchimento dell'uranio. Il worm sfruttava le vulnerabilità del sistema operativo Windows per ottenere l'accesso al software Siemens Step7 che controlla i PLC. Nonostante non sia un attacco diretto ai sistemi ML, Stuxnet è rilevante per tutti i sistemi embedded in quanto mostra il potenziale degli attori a livello statale per progettare attacchi che collegano il mondo informatico e quello fisico con effetti devastanti. Figura 14.1 spiega Stuxnet in modo più dettagliato.

Figura 14.1: Spiegazione di Stuxnet. Fonte: IEEE Spectrum

14.3.2 Hack della Jeep Cherokee

L'hack della Jeep Cherokee è stato un evento rivoluzionario che ha dimostrato i rischi insiti nelle automobili sempre più connesse (C. Miller 2019). In una dimostrazione controllata, i ricercatori della sicurezza hanno sfruttato da remoto una vulnerabilità nel sistema di entertainment Uconnect, che aveva una connessione cellulare a Internet. Sono stati in grado di controllare il motore, la trasmissione e i freni del veicolo, allarmando l'industria automobilistica e spingendola a riconoscere le gravi implicazioni per la sicurezza delle vulnerabilità informatiche nei veicoli. Video 14.1 di seguito è riportato un breve documentario dell'attacco.

Video 14.1: Hack della Jeep Cherokee

https://www.youtube.com/watch?v=MK0SrxBC1xs&ab_channel=WIRED

Sebbene non si sia trattato di un attacco a un sistema ML in sé, l'affidamento dei veicoli moderni ai sistemi embedded per funzioni critiche per la sicurezza presenta parallelismi significativi con l'implementazione di ML nei sistemi embedded, sottolineando la necessità di una sicurezza robusta a livello hardware.

14.3.3 Botnet Mirai

La botnet Mirai ha coinvolto l'infezione di dispositivi in rete come fotocamere digitali e lettori DVR (Antonakakis et al. 2017). Nell'ottobre 2016, la botnet è stata utilizzata per condurre uno dei più grandi attacchi DDoS, interrompendo l'accesso a Internet negli Stati Uniti. L'attacco è stato possibile perché molti dispositivi utilizzavano nomi utente e password predefiniti, che sono stati facilmente sfruttati dal malware Mirai per controllare i dispositivi. Video 14.2 spiega come funziona la botnet Mirai.

Video 14.2: Botnet Mirai

https://www.youtube.com/watch?v=1pywzRTJDaY

Sebbene i dispositivi non fossero basati su ML, l'incidente è un duro promemoria di ciò che può accadere quando numerosi dispositivi embedded con scarsi controlli di sicurezza vengono collegati in rete, cosa che sta diventando sempre più comune con la crescita dei dispositivi IoT basati su ML.

14.3.4 Implicazioni

Queste violazioni storiche dimostrano gli effetti a cascata delle vulnerabilità hardware nei sistemi embedded. Ogni incidente offre un precedente per comprendere i rischi e progettare protocolli di sicurezza migliori. Ad esempio, la botnet Mirai evidenzia l'immenso potenziale distruttivo quando gli autori delle minacce possono ottenere il controllo su dispositivi in rete con sicurezza debole, una situazione che sta diventando sempre più comune con i sistemi ML. Molti dispositivi ML attuali funzionano come dispositivi "edge" pensati per raccogliere ed elaborare dati localmente prima di inviarli al cloud. Proprio come le telecamere e i DVR compromessi da Mirai, i dispositivi ML edge spesso si basano su hardware embedded come processori ARM ed eseguono sistemi operativi leggeri come Linux. Proteggere le credenziali del dispositivo è fondamentale.

Allo stesso modo, l'hacking della Jeep Cherokee è stato un momento spartiacque per l'industria automobilistica. Ha esposto gravi vulnerabilità nei crescenti sistemi di veicoli connessi in rete e la loro mancanza di isolamento dai sistemi di guida principali come freni e sterzo. In risposta, i produttori di automobili hanno investito molto in nuove misure di sicurezza informatica, anche se probabilmente permangono delle lacune.

Chrysler ha effettuato un richiamo per correggere il software vulnerabile Uconnect, che consentiva l'exploit remoto. Ciò includeva l'aggiunta di protezioni a livello di rete per impedire l'accesso esterno non autorizzato e la compartimentazione dei sistemi di bordo per limitare i movimenti laterali. Sono stati aggiunti ulteriori livelli di crittografia per i comandi inviati tramite il bus CAN all'interno dei veicoli.

L'incidente ha anche stimolato la creazione di nuovi standard e best practice per la sicurezza informatica. L'Auto-ISAC è stato istituito per consentire alle case automobilistiche di condividere informazioni e la NHTSA ha guidato i rischi di gestione. Sono state sviluppate nuove procedure di test e audit per valutare le vulnerabilità in modo proattivo. Gli effetti collaterali continuano a guidare il cambiamento nel settore automobilistico poiché le auto diventano sempre più definite dal software.

Sfortunatamente, i produttori spesso trascurano la sicurezza quando sviluppano nuovi dispositivi edge ML, utilizzando password predefinite, comunicazioni non crittografate, aggiornamenti firmware non protetti, ecc. Tali vulnerabilità potrebbero consentire agli aggressori di ottenere l'accesso e controllare i dispositivi su larga scala infettandoli con malware. Con una botnet di dispositivi ML compromessi, gli aggressori potrebbero sfruttare la loro potenza di calcolo aggregata per attacchi DDoS su infrastrutture critiche.

Sebbene questi eventi non abbiano coinvolto direttamente hardware di machine learning, i principi degli attacchi si estendono ai sistemi ML, che spesso coinvolgono dispositivi embedded e architetture di rete simili. Poiché l'hardware ML è sempre più integrato con il mondo fisico, proteggerlo da tali violazioni è fondamentale. L'evoluzione delle misure di sicurezza in risposta a questi incidenti fornisce preziose informazioni sulla protezione dei sistemi ML attuali e futuri da vulnerabilità analoghe.

La natura distribuita dei dispositivi edge ML significa che le minacce possono propagarsi rapidamente attraverso le reti. E se i dispositivi vengono utilizzati per scopi critici come dispositivi medici, controlli industriali o veicoli a guida autonoma, il potenziale danno fisico dei bot ML armati potrebbe essere grave. Proprio come Mirai ha dimostrato il potenziale pericoloso dei dispositivi IoT scarsamente protetti, la prova del nove per la sicurezza dell'hardware ML sarà quanto questi dispositivi siano vulnerabili o resilienti ad attacchi simili a worm. La posta in gioco aumenta man mano che il ML si diffonde in ambiti critici per la sicurezza, ponendo l'onere sui produttori e sugli operatori di sistema di incorporare le lezioni di Mirai.

La lezione è l'importanza di progettare per la sicurezza fin dall'inizio e di avere difese stratificate. Il caso Jeep evidenzia potenziali vulnerabilità per i sistemi ML in merito alle interfacce software esterne e all'isolamento tra sottosistemi. I produttori di dispositivi e piattaforme ML dovrebbero assumere un approccio proattivo e completo simile alla sicurezza piuttosto che lasciarlo come un ripensamento. Una risposta rapida e la diffusione delle best practice saranno cruciali man mano che le minacce si evolvono.

14.4 Minacce alla Sicurezza per i Modelli ML

I modelli ML affrontano rischi per la sicurezza che possono comprometterne l'integrità, le prestazioni e l'affidabilità se non affrontati adeguatamente. Sebbene esistano diverse minacce, le principali includono: Furto di modelli, in cui gli avversari rubano i parametri proprietari del modello e i dati sensibili in essi contenuti. Avvelenamento dei dati, che compromette i modelli tramite manomissione dei dati. Gli attacchi avversari ingannano il modello per fare previsioni errate o indesiderate.

14.4.1 Furto di Modelli

Il furto di modelli si verifica quando un aggressore ottiene l'accesso non autorizzato a un modello ML distribuito. La preoccupazione in questo caso è il furto della struttura del modello e dei parametri addestrati, nonché dei dati proprietari in esso contenuti (Ateniese et al. 2015). Il furto di modelli è una minaccia reale e crescente, come dimostrato da casi come quello dell'ex ingegnere di Google Anthony Levandowski, che presumibilmente ha rubato i progetti di auto a guida autonoma di Waymo e ha fondato un'azienda concorrente. Oltre all'impatto economico, il furto di modelli può seriamente compromettere la privacy e consentire ulteriori attacchi.

Ad esempio, si consideri un modello ML sviluppato per raccomandazioni personalizzate in un'applicazione di e-commerce. Se un concorrente ruba questo modello, ottiene informazioni su analisi aziendali, preferenze dei clienti e persino segreti commerciali racchiusi nei dati del modello. Gli aggressori potrebbero sfruttare i modelli rubati per creare input più efficaci per attacchi di "inversione del modello", deducendo dettagli privati sui dati di addestramento del modello. Un modello di raccomandazione di e-commerce clonato potrebbe rivelare i comportamenti di acquisto e i dati demografici dei clienti.

Per comprendere gli attacchi di "inversione del modello", si consideri un sistema di riconoscimento facciale utilizzato per concedere l'accesso a strutture protette. Il sistema viene addestrato su un set di dati di foto dei dipendenti. Un aggressore potrebbe dedurre le caratteristiche del set di dati originale osservando l'output del modello su vari input. Ad

esempio, supponiamo che il livello di confidenza del modello per un particolare volto sia significativamente più alto per un dato set di caratteristiche. In tal caso, un aggressore potrebbe dedurre che qualcuno con quelle caratteristiche è probabile che sia nel set di dati di addestramento.
La metodologia di "inversione del modello" in genere prevede i seguenti passaggi:

- **Accesso agli Output del Modello:** L'aggressore interroga il modello ML con dati di input e osserva gli output. Ciò avviene spesso tramite un'interfaccia legittima, come un'API pubblica.
- **Analisi dei Confidence Score:** Per ogni input, il modello fornisce un "punteggio di confidenza" che riflette quanto l'input sia simile ai dati di training.
- **Reverse-Engineering:** Analizzando i punteggi di confidenza o le probabilità di output, gli aggressori possono utilizzare tecniche di ottimizzazione per ricostruire ciò che ritengono sia vicino ai dati di input originali.

Un esempio storico di tale vulnerabilità esplorata è stata la ricerca sugli attacchi di inversione contro il set di dati del premio Netflix degli Stati Uniti, in cui i ricercatori hanno dimostrato che era possibile conoscere le preferenze cinematografiche di un individuo, il che potrebbe portare a violazioni della privacy (Narayanan e Shmatikov 2006).
Il furto di modelli implica che potrebbe portare a perdite economiche, minare il vantaggio competitivo e violare la privacy degli utenti. C'è anche il rischio di attacchi di inversione del modello, in cui un avversario potrebbe immettere vari dati nel modello rubato per dedurre informazioni sensibili sui dati di addestramento.
In base alla risorsa desiderata, gli attacchi con furto di modelli possono essere suddivisi in due categorie: proprietà esatte del modello e comportamento approssimativo del modello.

14.4.1.0.1 Furto di Proprietà Esatte del Modello

In questi attacchi, l'obiettivo è estrarre informazioni su metriche concrete, come i parametri appresi di una rete, gli iperparametri ottimizzati e l'architettura interna dei layer del modello (Oliynyk, Mayer, e Rauber 2023).

- **Parametri Appresi:** Gli avversari mirano a rubare la conoscenza appresa di un modello (pesi e bias) per replicarla. Il furto di parametri è generalmente utilizzato con altri attacchi, come il furto di architettura, che non hanno conoscenza dei parametri.
- **Iperparametri Ottimizzati:** L'addestramento è costoso e l'identificazione della configurazione ottimale degli iperparametri (come velocità di apprendimento e regolarizzazione) può richiedere molto tempo e risorse. Di conseguenza, rubare gli iperparametri ottimizzati di un modello consente agli avversari di replicare il modello senza sostenere gli stessi costi di sviluppo.
- **Architettura del Modello:** Questo attacco riguarda la progettazione e la struttura specifiche del modello, come strati, neuroni e pattern di connettività. Oltre a ridurre i costi di training associati, questo furto rappresenta un grave rischio per la proprietà intellettuale, potenzialmente compromettendo il vantaggio competitivo di un'azienda. Il furto di architettura può essere ottenuto sfruttando attacchi side-channel (discussi più avanti).

14.4.1.0.2 Furto del Comportamento Approssimativo del Modello

Invece di estrarre valori numerici esatti dei parametri del modello, questi attacchi mirano a riprodurre il comportamento del modello (previsioni ed efficacia), il processo decisionale e le caratteristiche di alto livello (Oliynyk, Mayer, e Rauber 2023). Queste tecniche mirano a ottenere risultati simili pur consentendo deviazioni interne nei parametri e nell'architettura. I tipi di furto di comportamento approssimativo includono l'ottenimento dello stesso livello di efficacia e l'ottenimento di coerenza di previsione.

- **Livello di efficacia:** Gli aggressori mirano a replicare le capacità decisionali del modello piuttosto che concentrarsi sui valori precisi dei parametri. Ciò avviene attraverso la comprensione del comportamento complessivo del modello. Consideriamo uno scenario in cui un aggressore desidera copiare il comportamento di un modello di classificazione delle immagini. Analizzando i limiti decisionali del modello, l'attacco ottimizza il suo modello per raggiungere un'efficacia paragonabile al modello originale. Ciò potrebbe comportare l'analisi di 1) la matrice di confusione per comprendere l'equilibrio delle metriche di previsione (vero positivo, vero negativo, falso positivo, falso negativo) e 2) altre metriche di prestazione, come punteggio F1 e precisione, per garantire che i due modelli siano comparabili.
- **Coerenza della Previsione:** L'attaccante cerca di allineare i pattern di previsione del proprio modello con quelli del modello target. Ciò comporta l'abbinamento degli output di previsione (sia positivi che negativi) sullo stesso set di input e la garanzia della coerenza distributiva tra classi diverse. Ad esempio, prendiamo in considerazione un modello di elaborazione del linguaggio naturale (NLP) che genera un'analisi del sentiment per le recensioni di film (etichettando le recensioni come positive, neutre o negative). L'attaccante cercherà di mettere a punto il proprio modello per adattarlo alla previsione dei modelli originali sullo stesso set di recensioni di film. Ciò include la garanzia che il modello commetta gli stessi errori (previsioni errate) commessi dal modello target.

14.4.1.1 Caso di Studio

Nel 2018, Tesla ha intentato una causa contro la startup di auto a guida autonoma Zoox, sostenendo che ex dipendenti avevano rubato dati riservati e segreti commerciali relativi al sistema di assistenza alla guida autonoma di Tesla.
Tesla ha affermato che diversi suoi ex dipendenti hanno sottratto oltre 10 GB. di dati proprietari, inclusi modelli ML e codice sorgente, prima di unirsi a Zoox. Ciò avrebbe incluso uno dei modelli di riconoscimento delle immagini cruciali di Tesla per l'identificazione degli oggetti.
Il furto di questo modello proprietario sensibile potrebbe aiutare Zoox ad abbreviare anni di sviluppo ML e duplicare le capacità di Tesla. Tesla ha sostenuto che questo furto di I.P. ha causato danni finanziari e competitivi significativi. C'erano anche preoccupazioni che potesse consentire attacchi di inversione del modello per dedurre dettagli privati sui dati di test di Tesla.
I dipendenti di Zoox hanno negato di aver rubato informazioni proprietarie. Tuttavia, il caso evidenzia i rischi significativi del furto di modelli, che consente la clonazione di modelli commerciali, causando ripercussioni economiche e aprendo la porta a ulteriori violazioni della privacy dei dati.

14.4.2 Avvelenamento dei Dati

L'avvelenamento dei dati è un attacco in cui i dati di training vengono manomessi, portando a un modello compromesso (Biggio, Nelson, e Laskov 2012). Gli aggressori possono modificare gli esempi di training esistenti, inserire nuovi dati dannosi o influenzare il processo di raccolta dati. I dati avvelenati vengono etichettati in modo tale da alterare il comportamento appreso del modello. Ciò può essere particolarmente dannoso nelle applicazioni in cui i modelli ML prendono decisioni automatizzate in base a pattern appresi. Oltre ai set di training, i test di avvelenamento e i dati di convalida possono consentire agli avversari di aumentare artificialmente le prestazioni del modello segnalate.
Il processo di solito prevede i seguenti passaggi:

- **Injection:** L'aggressore aggiunge esempi errati o fuorvianti al set di training. Questi esempi sono spesso progettati per sembrare normali a un'ispezione superficiale, ma sono stati attentamente elaborati per interrompere il processo di apprendimento.
- **Training:** Il modello ML si allena su questo set di dati manipolato e sviluppa comprensioni distorte dei pattern di dati.
- **Deployment:** Una volta distribuito il modello, l'addestramento corrotto porta a un processo decisionale imperfetto o a vulnerabilità prevedibili che l'aggressore può sfruttare.

Gli impatti dell'avvelenamento dei dati vanno oltre i semplici errori di classificazione o cali di accuratezza. Ad esempio, se dati errati o dannosi vengono introdotti nel set di addestramento di un sistema di riconoscimento dei segnali stradali, il modello potrebbe imparare a classificare erroneamente i segnali di stop come segnali di precedenza, il che può avere pericolose conseguenze nel mondo reale, specialmente nei sistemi autonomi embedded come i veicoli autonomi.

L'avvelenamento dei dati può degradare l'accuratezza di un modello, costringerlo a fare previsioni errate o farlo comportare in modo imprevedibile. In applicazioni critiche come l'assistenza sanitaria, tali alterazioni possono portare a significativi problemi di fiducia e sicurezza.

Esistono sei categorie principali di avvelenamento dei dati (Oprea, Singhal, e Vassilev 2022):

- **Attacchi di Disponibilità:** Questi attacchi cercano di compromettere la funzionalità complessiva di un modello. Fanno sì che classifichi erroneamente la maggior parte dei campioni di test, rendendo il modello inutilizzabile per applicazioni pratiche. Un esempio è il "label flipping", in cui le etichette di una classe specifica e mirata vengono sostituite con etichette di una classe diversa.
- **Attacchi Mirati:** A differenza degli attacchi alla disponibilità, gli attacchi mirati mirano a compromettere un numero limitato di campioni di test. Quindi, l'effetto è localizzato su un numero limitato di classi, mentre il modello mantiene lo stesso livello di accuratezza originale sulla maggior parte delle classi. La natura mirata dell'attacco richiede che l'aggressore conosca le classi del modello, rendendo più difficile il rilevamento di questi attacchi.
- **Attacchi Backdoor:** In questi attacchi, un avversario prende di mira pattern specifici nei dati. L'aggressore introduce una backdoor (un trigger o pattern nascosto e dannoso) nei dati di training, ad esempio modificando determinate feature nei dati strutturati o un pattern di pixel in una posizione fissa. Ciò fa sì che il modello associ il pattern dannoso a etichette specifiche. Di conseguenza, quando il modello incontra campioni di test che contengono un pattern dannoso, fa false previsioni, evidenziando l'importanza della cautela e della prevenzione nel ruolo dei professionisti della sicurezza dei dati.
- **Attacchi di Sotto-popolazione:** Gli aggressori scelgono selettivamente di compromettere un sottoinsieme dei campioni di test mantenendo l'accuratezza sul resto dei campioni. Questi attacchi si possono pensare come una combinazione di attacchi di disponibilità e mirati: eseguire attacchi di disponibilità (degrado delle prestazioni) nell'ambito di un sottoinsieme mirato. Sebbene gli attacchi di sottopopolazione possano sembrare molto simili agli attacchi mirati, i due presentano chiare differenze:
- **Scope:** Mentre gli attacchi mirati prendono di mira un set selezionato di campioni, gli attacchi di sotto-popolazione prendono di mira una sotto-popolazione generale con rappresentazioni di caratteristiche simili. Ad esempio, in un attacco mirato, un attore inserisce immagini manipolate di un cartello di avvertimento "rallentamenti" (con

perturbazioni o pattern attentamente studiati), che fa sì che un'auto autonoma non riconosca tale cartello e non rallenti. D'altro canto, manipolare tutti i campioni di persone con accento britannico in modo che un modello di riconoscimento vocale classifichi erroneamente il discorso di una persona britannica è un esempio di attacco di sotto-popolazione.

- **Conoscenza:** Mentre gli attacchi mirati richiedono un alto grado di familiarità con i dati, gli attacchi alla sotto-popolazione richiedono una conoscenza meno approfondita per essere efficaci.

14.4.2.1 Caso di Studio 1

Nel 2017, i ricercatori hanno dimostrato un attacco di avvelenamento dei data contro un modello di classificazione della tossicità popolare chiamato Perspective (Hosseini et al. 2017). Questo modello ML rileva commenti tossici online.

I ricercatori hanno aggiunto commenti tossici generati sinteticamente con lievi errori di ortografia e grammaticali ai dati di training del modello. Ciò ha lentamente corrotto il modello, facendogli classificare erroneamente un numero crescente di input gravemente tossici come non tossici nel tempo.

Dopo il ri-addestramento sui dati avvelenati, il tasso di falsi negativi del modello è aumentato dall'1,4% al 27%, consentendo ai commenti estremamente tossici di aggirare il rilevamento. I ricercatori hanno avvertito che questo furtivo "data poisoning" potrebbe consentire la diffusione di discorsi di odio, molestie e abusi se implementato contro sistemi di moderazione reali.

Questo caso evidenzia come l'avvelenamento dei dati possa degradare l'accuratezza e l'affidabilità del modello. Per le piattaforme di social media, un attacco di avvelenamento che compromette il rilevamento della tossicità potrebbe portare alla proliferazione di contenuti dannosi e alla sfiducia nei sistemi di moderazione ML. L'esempio dimostra perché proteggere l'integrità dei dati di training e monitorare l'avvelenamento è fondamentale in tutti i domini applicativi.

14.4.2.2 Caso di Studio 2

È interessante notare che gli attacchi di "data poisoning" non sono sempre dannosi (Shan et al. 2023). Nightshade, uno strumento sviluppato da un team guidato dal professor Ben Zhao presso l'Università di Chicago, utilizza il data poisoning per aiutare gli artisti a proteggere la propria arte da scraping e violazioni del copyright da parte di modelli di intelligenza artificiale generativa. Gli artisti possono utilizzare lo strumento per modificare le proprie immagini in modo sottile prima di caricarle online.

Sebbene queste modifiche siano impercettibili all'occhio umano, possono degradare significativamente le prestazioni dei modelli di intelligenza artificiale generativa quando integrate nei dati di addestramento. I modelli generativi possono essere manipolati per produrre output irrealistici o privi di senso. Ad esempio, con solo 300 immagini corrotte, i ricercatori dell'Università di Chicago sono riusciti a ingannare l'ultimo modello "Stable Diffusion" per generare immagini di cani che assomigliano a felini o bovini quando richiesto per le automobili.

Con l'aumento della quantità di immagini corrotte online, l'efficacia dei modelli addestrati su dati estratti diminuirà esponenzialmente. Inizialmente, identificare i dati corrotti è difficile e richiede un intervento manuale. Successivamente, la contaminazione si diffonde rapidamente ai concetti correlati, poiché i modelli generativi stabiliscono connessioni tra le parole e le loro rappresentazioni visive. Di conseguenza, un'immagine corrotta di un'"auto" potrebbe propagarsi in immagini generate collegate a termini come "camion", "treno" e "autobus".

D'altro canto, questo strumento può essere utilizzato in modo dannoso e influenzare le applicazioni legittime del modello generativo. Ciò dimostra la natura molto impegnativa e innovativa degli attacchi di apprendimento automatico.
Figura 17.26 mostra gli effetti di diversi livelli di avvelenamento dei dati (50 campioni, 100 campioni e 300 campioni di immagini avvelenate) sulla generazione di immagini in varie categorie. Notare come le immagini iniziano a deformarsi e deviare dalla categoria desiderata. Ad esempio, dopo 300 campioni di avvelenamento, una richiesta di un'auto genera una mucca.

COURTESY OF THE RESEARCHERS

Figura 14.2: Avvelenamento dei Dati. Fonte: Shan et al. (2023).

14.4.3 Attacchi Avversari

Gli attacchi avversari mirano a indurre i modelli a fare previsioni errate fornendo loro input ingannevoli appositamente creati (chiamati esempi avversari) (Parrish et al. 2023). Aggiungendo lievi perturbazioni ai dati di input, gli avversari possono "hackerare" il riconoscimento di pattern di un modello e ingannarlo. Si tratta di tecniche sofisticate in cui lievi, spesso impercettibili alterazioni dei dati di input possono indurre un modello ML a fare una previsione errata.
È possibile generare prompt che portano a immagini non sicure in modelli testo-immagine come DALLE (Ramesh et al. 2021) o Stable Diffusion (Rombach et al. 2022). Ad esempio, alterando i valori dei pixel di un'immagine, gli aggressori possono ingannare un sistema di riconoscimento facciale facendogli identificare un volto come una persona diversa.
Gli attacchi avversari sfruttano il modo in cui i modelli ML apprendono e prendono decisioni durante l'inferenza. Questi modelli funzionano sul principio di riconoscimento di pattern nei dati. Un avversario crea input dannosi con perturbazioni per fuorviare il riconoscimento di pattern del modello, essenzialmente "hackerando" le percezioni del modello.
Gli attacchi avversari rientrano in diversi scenari:

- **Attacchi Whitebox:** L'attaccante ha una conoscenza completa del funzionamento interno del modello target, inclusi i dati di addestramento, i parametri e l'architettura. Questo ampio accesso facilita lo sfruttamento delle vulnerabilità del modello. L'attaccante può sfruttare debolezze specifiche e sottili per costruire esempi avversari altamente efficaci.
- **Attacchi Blackbox:** A differenza degli attacchi Whitebox, in quelli Blackbox l'attaccante ha poca o nessuna conoscenza del modello target. L'attore avversario

deve osservare attentamente il comportamento di output del modello per eseguire l'attacco.

- **Attacchi Greybox:** Questi attacchi occupano uno spettro tra gli attacchi Blackbox e Whitebox. L'avversario possiede una conoscenza parziale della struttura interna del modello target. Ad esempio, l'attaccante potrebbe conoscere i dati di training ma non avere informazioni sull'architettura o sui parametri del modello. In scenari pratici, la maggior parte degli attacchi rientra in questa zona grigia.

Il panorama dei modelli di apprendimento automatico è complesso e ampio, soprattutto data la loro relativamente recente integrazione nelle applicazioni commerciali. Questa rapida adozione, sebbene trasformativa, ha portato alla luce numerose vulnerabilità all'interno di questi modelli. Di conseguenza, sono emersi vari metodi di attacco avversari, ognuno dei quali sfrutta strategicamente diversi aspetti di vari modelli. Di seguito, evidenziamo un sottoinsieme di questi metodi, che mostra la natura multiforme degli attacchi avversari sui modelli di apprendimento automatico:

- I **Generative Adversarial Network (GAN)** sono modelli di deep learning costituiti da due reti in competizione tra loro: un generatore e un discriminatore (Goodfellow et al. 2020). Il generatore cerca di sintetizzare dati realistici mentre il discriminatore valuta se sono reali o falsi. Le GAN possono essere utilizzate per creare esempi avversari. La rete del generatore è addestrata per produrre input che il modello target classifica erroneamente. Queste immagini generate da GAN possono quindi attaccare un classificatore target o un modello di rilevamento. Il generatore e il modello target sono impegnati in un processo competitivo, con il generatore che migliora continuamente la sua capacità di creare esempi ingannevoli e il modello target che aumenta la sua resistenza a tali esempi. Le GAN forniscono un framework robusto per creare input avversari complessi e diversi, illustrando l'adattabilità dei modelli generativi nel panorama avversario.
- I **Transfer Learning Adversarial Attacks** [attacchi avversari di apprendimento di trasferimento] sfruttano la conoscenza trasferita da un modello pre-addestrato a un modello target, creando esempi avversari che possono ingannare entrambi i modelli. Questi attacchi rappresentano una preoccupazione crescente, in particolare quando gli avversari hanno conoscenza dell'estrattore di feature ma non hanno accesso alla testa di classificazione (la parte o il layer responsabile della creazione delle classificazioni finali). Denominate "attacchi headless", queste strategie avversarie trasferibili sfruttano le capacità espressive degli estrattori di feature per creare perturbazioni, ignare dello spazio delle etichette o dei dati di training. L'esistenza di tali attacchi sottolinea l'importanza di sviluppare difese robuste per le applicazioni di apprendimento tramite trasferimento, soprattutto perché i modelli pre-addestrati sono comunemente utilizzati (Abdelkader et al. 2020).

14.4.3.1 Caso di Studio

Nel 2017, i ricercatori hanno condotto esperimenti posizionando piccoli adesivi bianchi e neri sui segnali di stop (Eykholt et al. 2017). Quando visti da un occhio umano normale, gli adesivi non oscuravano il segnale né ne impedivano l'interpretazione. Tuttavia, quando le immagini degli adesivi dei segnali di stop venivano inserite nei modelli ML standard di classificazione dei segnali stradali, venivano classificati erroneamente come segnali di limite di velocità nell'85% dei casi.

Questa dimostrazione ha mostrato come semplici adesivi avversari potrebbero ingannare i sistemi ML facendogli interpretare male i segnali stradali critici. Se implementati in modo realistico, questi attacchi potrebbero mettere a repentaglio la sicurezza pubblica, inducendo i veicoli autonomi a interpretare male i segnali di stop come limiti di velocità. I ricercatori

hanno avvertito che ciò potrebbe potenzialmente causare pericolosi semplici rallentamenti o accelerazioni negli incroci.
Questo caso di studio fornisce un'illustrazione concreta di come gli esempi avversari sfruttano i meccanismi di riconoscimento di pattern dei modelli ML. Alterando in modo sottile i dati di input, gli aggressori possono indurre previsioni errate e rappresentare rischi significativi per applicazioni critiche per la sicurezza come le auto a guida autonoma. La semplicità dell'attacco dimostra come anche cambiamenti minori e impercettibili possano sviare i modelli. Di conseguenza, gli sviluppatori devono implementare difese robuste contro tali minacce.

14.5 Minacce alla Sicurezza Per l'Hardware ML

Un esame sistematico delle minacce alla sicurezza dell'hardware di apprendimento automatico embedded è essenziale per comprendere in modo completo le potenziali vulnerabilità nei sistemi ML. Inizialmente, verranno esplorate le vulnerabilità hardware derivanti da difetti di progettazione intrinseci che possono essere sfruttati. Questa conoscenza di base è fondamentale per riconoscere le origini delle debolezze hardware. Successivamente, verranno esaminati gli attacchi fisici, che rappresentano i metodi più diretti e palesi per compromettere l'integrità hardware. Sulla base di ciò, verranno analizzati gli attacchi di "injection" di guasti, dimostrando come le manipolazioni deliberate possano indurre guasti del sistema.
Passando agli attacchi "side-channel", verrà mostrata la crescente complessità, poiché questi si basano sullo sfruttamento di "leakage" [perdite] di informazioni indirette, che richiedono una comprensione sfumata delle operazioni hardware e delle interazioni ambientali. Le interfacce che "perdono" mostreranno come i canali di comunicazione esterni possono diventare vulnerabili, portando a esposizioni accidentali di dati. Le discussioni sull'hardware contraffatto traggono vantaggio dalle precedenti esplorazioni dell'integrità dell'hardware e dalle tecniche di sfruttamento, poiché spesso aggravano questi problemi con rischi aggiuntivi dovuti alla loro provenienza discutibile. Infine, i rischi della "supply chain" comprendono tutte le preoccupazioni di cui sopra e le inquadrano nel contesto del percorso dell'hardware dalla produzione alla distribuzione, evidenziando la natura multiforme della sicurezza dell'hardware e la necessità di vigilanza in ogni fase.
Tabella 14.1 riassume gli argomenti:

Tabella 14.1: Tipi di minaccia alla sicurezza hardware.

Tipo di minaccia	Descrizione	Rilevanza per la sicurezza hardware ML
Bug Hardware	Difetti intrinseci nelle progettazioni hardware che possono compromettere l'integrità del sistema.	Fondamento della vulnerabilità hardware.
Attacchi Fisici	Sfruttamento diretto dell'hardware tramite accesso fisico o manipolazione.	Modello di minaccia basilare e palese.
Attacchi di Injection di Guasti	Induzione di guasti per causare errori nel funzionamento dell'hardware, portando a potenziali crash di sistema.	Manipolazione sistematica che porta al guasto.
Attacchi a Canale Laterale	Sfruttamento di informazioni sul funzionamento dell'hardware per estrarre dati sensibili.	Attacco indiretto tramite osservazione ambientale.
Interfacce con Perdite	Vulnerabilità derivanti da interfacce che espongono i dati in modo involontario.	Esposizione dei dati tramite canali di comunicazione.
Hardware Contraffatto	Utilizzo di componenti hardware non autorizzati che potrebbero presentare falle di sicurezza.	Problemi di vulnerabilità aggravati.
Rischi della Catena di Fornitura	Rischi introdotti durante il ciclo di vita dell'hardware, dalla produzione alla distribuzione.	Sfide di sicurezza cumulative e multiformi.

14.5.1 Bug Hardware

L'hardware non è immune al problema pervasivo di difetti di progettazione o bug. Gli aggressori possono sfruttare queste vulnerabilità per accedere, manipolare o estrarre dati sensibili, violando la riservatezza e l'integrità da cui dipendono utenti e servizi. Un esempio di tali vulnerabilità è venuto alla luce con la scoperta di Meltdown e Spectre, due vulnerabilità hardware che sfruttano vulnerabilità critiche nei processori moderni. Questi bug consentono agli aggressori di aggirare la barriera hardware che separa le applicazioni, consentendo a un programma dannoso di leggere la memoria di altri programmi e del sistema operativo. Meltdown (Kocher et al. 2019a) e Spectre (Kocher et al. 2019b) funzionano sfruttando le ottimizzazioni nelle CPU moderne che consentono loro di eseguire istruzioni speculative fuori ordine prima che i controlli di validità siano stati completati. Ciò rivela dati che dovrebbero essere inaccessibili, che l'attacco cattura tramite canali laterali come le cache. La complessità tecnica dimostra la difficoltà di eliminare le vulnerabilità anche con una validazione estesa.

Se un sistema ML elabora dati sensibili, come informazioni personali degli utenti o analisi aziendali proprietarie, Meltdown e Spectre rappresentano un pericolo reale e presente per la sicurezza dei dati. Si consideri il caso di una scheda acceleratrice ML progettata per accelerare i processi di apprendimento automatico, come quelli di cui abbiamo parlato nel capitolo I.A. Hardware. Questi acceleratori lavorano con la CPU per gestire calcoli complessi, spesso correlati all'analisi dei dati, al riconoscimento delle immagini e all'elaborazione del linguaggio naturale. Se una scheda acceleratrice di questo tipo presenta una vulnerabilità simile a Meltdown o Spectre, potrebbe far trapelare i dati che elabora. Un aggressore potrebbe sfruttare questa falla non solo per sottrarre dati, ma anche per ottenere informazioni sul funzionamento del modello ML, incluso potenzialmente il reverse engineering del modello stesso (tornando quindi al problema del furto di modelli.

Uno scenario reale in cui ciò potrebbe essere devastante sarebbe nel settore sanitario. I sistemi ML elaborano regolarmente dati altamente sensibili dei pazienti per aiutare a diagnosticare, pianificare il trattamento e prevedere i risultati. Un bug nell'hardware del sistema potrebbe portare alla divulgazione non autorizzata di informazioni sanitarie personali, violando la privacy del paziente e contravvenendo a rigidi standard normativi come l'Health Insurance Portability and Accountability Act (HIPAA)

Le vulnerabilità Meltdown e Spectre sono un duro promemoria del fatto che la sicurezza hardware non riguarda solo la prevenzione dell'accesso fisico non autorizzato, ma anche la garanzia che l'architettura dell'hardware non diventi un canale per l'esposizione dei dati. Difetti di progettazione hardware simili emergono regolarmente in CPU, acceleratori, memoria, bus e altri componenti. Ciò richiede continue mitigazioni retroattive e compromessi sulle prestazioni nei sistemi distribuiti. Soluzioni proattive come le architetture di elaborazione confidenziale potrebbero mitigare intere classi di vulnerabilità attraverso una progettazione hardware fondamentalmente più sicura. Contrastare i bug hardware richiede rigore in ogni fase di progettazione, validazione e distribuzione.

14.5.2 Attacchi Fisici

La manomissione fisica si riferisce alla manipolazione diretta e non autorizzata di risorse informatiche fisiche per minare l'integrità dei sistemi di apprendimento automatico. È un attacco particolarmente insidioso perché aggira le tradizionali misure di sicurezza informatica, che spesso si concentrano più sulle vulnerabilità del software che sulle minacce hardware.

La manomissione fisica può assumere molte forme, da quelle relativamente semplici, come l'inserimento di un dispositivo USB caricato con software dannoso in un server, a quelle altamente sofisticate, come l'inclusione di un Trojan hardware durante il processo di

produzione di un microchip (discusso più avanti in dettaglio nella sezione Supply Chain). I sistemi ML sono suscettibili a questo attacco perché si basano sull'accuratezza e l'integrità del loro hardware per elaborare e analizzare correttamente grandi quantità di dati.
Si consideri un drone alimentato da ML utilizzato per la mappatura geografica. Il funzionamento del drone si basa su una serie di sistemi di bordo, tra cui un modulo di navigazione che elabora gli input da vari sensori per determinare il suo percorso. Se un aggressore ottiene l'accesso fisico a questo drone, potrebbe sostituire il modulo di navigazione originale con uno compromesso che include una backdoor. Questo modulo manipolato potrebbe quindi alterare la traiettoria di volo del drone per condurre la sorveglianza su aree riservate o persino contrabbandare merci di contrabbando volando su rotte non rilevate.
Un altro esempio è la manomissione fisica degli scanner biometrici utilizzati per il controllo degli accessi in strutture sicure. Introducendo un sensore modificato che trasmette dati biometrici a un ricevitore non autorizzato, un aggressore può accedere ai dati di identificazione personale per autenticare gli individui.
Esistono diversi modi in cui la manomissione fisica può verificarsi nell'hardware ML:

- **Manipolazione dei sensori:** Si consideri un veicolo autonomo dotato di telecamere e LiDAR per la percezione ambientale. Un malintenzionato potrebbe manipolare deliberatamente l'allineamento fisico di questi sensori per creare zone di occlusione o distorcere le misure della distanza. Ciò potrebbe compromettere le capacità di rilevamento degli oggetti e potenzialmente mettere in pericolo gli occupanti del veicolo.
- **Trojan hardware:** Le modifiche dannose ai circuiti possono introdurre trojan progettati per attivarsi in base a specifiche condizioni di input. Ad esempio, un chip acceleratore ML potrebbe funzionare come previsto fino a quando non incontra un trigger predeterminato, dopodiché si comporta in modo irregolare.
- **Manomissione della memoria:** L'esposizione fisica e la manipolazione dei chip di memoria potrebbero consentire l'estrazione di parametri del modello ML crittografati. Le tecniche di iniezione di guasti possono anche corrompere i dati del modello per degradare l'accuratezza.
- **Introduzione di backdoor:** Ottenendo l'accesso fisico ai server, un avversario potrebbe utilizzare keylogger hardware per catturare password e creare account backdoor per l'accesso persistente. Questi potrebbero poi essere utilizzati per esfiltrare dati di training ML nel tempo.
- **Attacchi alla supply chain:** Manipolare componenti hardware di terze parti o compromettere i canali di produzione e spedizione crea vulnerabilità sistemiche difficili da rilevare e correggere.

14.5.3 Attacchi di Fault-injection

Introducendo intenzionalmente guasti nell'hardware ML, gli aggressori possono indurre errori nel processo di elaborazione, portando a output non corretti. Questa manipolazione compromette l'integrità delle operazioni ML e può fungere da vettore per ulteriori sfruttamenti, come il reverse engineering del sistema o il bypass del protocollo di sicurezza.
L'iniezione di guasti comporta l'interruzione deliberata delle operazioni di elaborazione standard in un sistema tramite interferenze esterne (Joye e Tunstall 2012). Attivando con precisione gli errori di elaborazione, gli avversari possono alterare l'esecuzione del programma in modi che degradano l'affidabilità o trapelano informazioni sensibili.
Per l'iniezione di guasti possono essere utilizzate varie tecniche di manomissione fisica. Bassa tensione (Barenghi et al. 2010), picchi di potenza (Hutter, Schmidt, e Plos 2009), anomalie di clock (Amiel, Clavier, e Tunstall 2006), impulsi elettromagnetici (Agrawal et al.

2007), aumento della temperatura (S. Skorobogatov 2009) e colpi laser (S. P. Skorobogatov e Anderson 2003) sono comuni vettori di attacco hardware. Sono programmati con precisione per indurre guasti come bit invertiti o istruzioni saltate durante operazioni critiche.
Per i sistemi ML, le conseguenze includono una precisione del modello compromessa, negazione del servizio, estrazione di dati di training privati o parametri del modello e reverse engineering delle architetture del modello. Gli aggressori potrebbero utilizzare l'iniezione di guasti per forzare classificazioni errate, interrompere sistemi autonomi o rubare proprietà intellettuale.
Ad esempio, Breier et al. (2018) ha iniettato con successo un "fault attack" in una rete neurale profonda distribuita su un microcontrollore. Hanno utilizzato un laser per riscaldare transistor specifici, costringendoli a cambiare stato. In un caso, hanno utilizzato questo metodo per attaccare una funzione di attivazione ReLU, con il risultato che la funzione emetteva sempre un valore di 0, indipendentemente dall'input. Nel codice assembly mostrato in Figura 14.3, l'attacco ha fatto sì che il programma in esecuzione saltasse sempre l'istruzione `jmp end` alla riga 6. Ciò significa che `HiddenLayerOutput[i]` è sempre impostato su 0, sovrascrivendo eventuali valori scritti su di esso nelle righe 4 e 5. Di conseguenza, i neuroni mirati vengono resi inattivi, con conseguenti classificazioni errate.

```
1          ldi r1, 0        ;load 0 to r1
2          cp r1, r15       ;compare MSB of Accum to r1
3          brge else        ;jump to else if 0 >= Accum
4          movw r10, r15    ;HiddenLayerOutput[i] = Accum
5          movw r12, r17    ;HiddenLayerOutput[i] = Accum
6          jmp end          ;jump after the else statement
7  else:   clr r10          ;HiddenLayerOutput[i]= 0
8          clr r11          ;HiddenLayerOutput[i]= 0
9          clr r12          ;HiddenLayerOutput[i]= 0
10         clr r13          ;HiddenLayerOutput[i]= 0
11 end:    ...              ;continue the execution
```

Figura 14.3: Iniezione di errore dimostrata con codice assembly. Fonte: Breier et al. (2018).

La strategia di un aggressore potrebbe essere quella di dedurre informazioni sulle funzioni di attivazione tramite attacchi side-channel (discussi in seguito). Quindi, l'aggressore potrebbe tentare di colpire più calcoli di funzioni di attivazione iniettando casualmente guasti nei livelli il più vicino possibile al livello di output, aumentando la probabilità e l'impatto dell'attacco.
I dispositivi embedded sono particolarmente vulnerabili a causa di un limitato rafforzamento fisico e di vincoli di risorse che limitano le difese di runtime robuste. Senza un packaging antimanomissione, l'accesso dell'aggressore ai bus di sistema e alla memoria consente di infierire guasti precisi. Anche i modelli ML embedded leggeri mancano di ridondanza per bypassare gli errori.
Questi attacchi possono essere particolarmente insidiosi perché aggirano le tradizionali misure di sicurezza basate su software, spesso non tenendo conto delle interruzioni fisiche. Inoltre, poiché i sistemi ML si basano in larga misura sull'accuratezza e l'affidabilità del loro hardware per attività come il riconoscimento di pattern, il processo decisionale e le risposte automatiche, qualsiasi compromesso nel loro funzionamento dovuto all'iniezione di guasti può avere conseguenze gravi e di vasta portata.
Per mitigare i rischi di iniezione di guasti è necessario un approccio multi-layer. Il rafforzamento fisico tramite custodie antimanomissione e offuscamento del design aiuta a ridurre l'accesso. Il rilevamento di leggere anomalie può identificare input di sensori insoliti o output di modelli errati (Hsiao et al. 2023). Le memorie con correzione degli errori riducono al minimo le interruzioni, mentre la crittografia dei dati salvaguarda le informazioni. Le tecniche emergenti di watermarking dei modelli tracciano i parametri rubati.

Tuttavia, bilanciare protezioni robuste con i limiti di dimensioni e potenza ristretti dei sistemi embedded rimane una sfida. I limiti della crittografia e la mancanza di coprocessori sicuri su hardware embedded sensibile ai costi limitano le opzioni. In definitiva, la resilienza all'iniezione di guasti richiede una prospettiva multi-layer che abbraccia i layer di progettazione elettrica, firmware, software e fisica.

14.5.4 Attacchi a canale laterale

Gli attacchi side-channel costituiscono una classe di violazioni della sicurezza che sfruttano informazioni rivelate inavvertitamente tramite l'implementazione fisica dei sistemi informatici. Contrariamente agli attacchi diretti che prendono di mira vulnerabilità software o di rete, questi attacchi sfruttano le caratteristiche hardware intrinseche del sistema per estrarre informazioni sensibili.

La premessa fondamentale di un attacco side-channel è che il funzionamento di un dispositivo può rivelare inavvertitamente informazioni. Tali fughe possono provenire da varie fonti, tra cui l'energia elettrica consumata da un dispositivo (Kocher, Jaffe, e Jun 1999), i campi elettromagnetici che emette (Gandolfi, Mourtel, e Olivier 2001), il tempo necessario per elaborare determinate operazioni o persino i suoni che produce. Ogni canale può intravedere indirettamente i processi interni del sistema, rivelando informazioni che possono compromettere la sicurezza.

Ad esempio, si consideri un sistema di apprendimento automatico che esegue transazioni crittografate. Gli algoritmi di crittografia dovrebbero proteggere i dati, ma richiedono un lavoro computazionale per crittografare e decrittografare le informazioni. Un aggressore può analizzare i pattern di consumo energetico del dispositivo che esegue la crittografia per scoprire la chiave crittografica. Con metodi statistici sofisticati, piccole variazioni nel consumo energetico durante il processo di crittografia possono essere correlate ai dati in fase di elaborazione, rivelando infine la chiave. Alcune tecniche di attacco di analisi differenziale sono "Differential Power Analysis (DPA)" (Kocher et al. 2011), "Differential Electromagnetic Analysis (DEMA)" e "Correlation Power Analysis (CPA)".

Ad esempio, si consideri un aggressore che cerca di violare l'algoritmo di crittografia AES utilizzando un attacco di analisi differenziale. L'aggressore dovrebbe prima raccogliere molte tracce di potenza o elettromagnetiche (una traccia è una registrazione di consumi o emissioni) del dispositivo durante l'esecuzione della crittografia AES.

Una volta che l'aggressore ha raccolto tracce sufficienti, utilizzerebbe una tecnica statistica per identificare le correlazioni tra le tracce e i diversi valori del testo in chiaro (testo originale non crittografato) e del testo cifrato (testo crittografato). Queste correlazioni verrebbero poi utilizzate per dedurre il valore di un bit nella chiave AES e, infine, l'intera chiave. Gli attacchi di analisi differenziale sono pericolosi perché sono economici, efficaci e non intrusivi, consentendo agli aggressori di aggirare le misure di sicurezza algoritmiche e a livello hardware. I sistemi compromessi da questi attacchi sono anche difficili da rilevare perché non modificano fisicamente il dispositivo o interrompono l'algoritmo di crittografia.

Di seguito, una visualizzazione semplificata illustra come l'analisi dei pattern di consumo energetico del dispositivo di crittografia può aiutare a estrarre informazioni sulle operazioni dell'algoritmo e, a sua volta, sui dati segreti. Consideriamo un dispositivo che accetta una password di 5 byte come input. I diversi pattern di tensione misurati mentre il dispositivo di crittografia esegue operazioni sull'input per autenticare la password verranno analizzati e confrontati.

Innanzitutto, l'analisi della potenza elettrica delle operazioni del dispositivo dopo aver inserito una password corretta è mostrata nella prima immagine in Figura 14.4. Il grafico blu denso restituisce la misura della tensione del dispositivo di crittografia. Ciò che è

significativo qui è il confronto tra i diversi grafici di analisi piuttosto che i dettagli specifici di ciò che sta accadendo in ogni scenario.

Figura 14.4: Analisi della potenza di un dispositivo di crittografia con una password corretta. Fonte: Colin O'Flynn.

Quando viene inserita una password errata, il grafico dell'analisi della potenza è mostrato in Figura 14.5. I primi tre byte della password sono corretti. Di conseguenza, i pattern di tensione sono molto simili o identici tra i due grafici, fino al quarto byte incluso. Dopo che il dispositivo elabora il quarto byte, viene determinata una mancata corrispondenza tra la chiave segreta e l'input tentato. Viene notato un cambiamento nel pattern nel punto di transizione tra il quarto e il quinto byte: la tensione aumenta (la corrente diminuisce) perché il dispositivo ha smesso di elaborare il resto dell'input.

Figura 14.5: Analisi della potenza di un dispositivo di crittografia con una password (parzialmente) errata. Fonte: Colin O'Flynn.

Figura 14.6 descrive un altro grafico di una password completamente errata. Dopo che il dispositivo ha terminato l'elaborazione del primo byte, determina che è errato e interrompe l'ulteriore elaborazione: la tensione aumenta e la corrente diminuisce.

Figura 14.6: Analisi della potenza di un dispositivo di crittografia con una password errata. Fonte: Colin O'Flynn.

L'esempio sopra dimostra come le informazioni sul processo di crittografia e sulla chiave segreta possono essere dedotte analizzando diversi input e tentando di "intercettare" le operazioni del dispositivo su ogni byte di input. Per una spiegazione più dettagliata, guardare Video 14.3 di seguito.

Video 14.3: Power Attack

https://www.youtube.com/watch?v=2iDLfuEBcs8

Un altro esempio è un sistema ML per il riconoscimento vocale, che elabora i comandi vocali per eseguire azioni. Misurando la latenza del sistema per rispondere ai comandi o la potenza utilizzata durante l'elaborazione, un aggressore potrebbe dedurre quali comandi vengono elaborati e quindi apprendere i pattern operativi del sistema. Ancora più sottilmente, il suono emesso dalla ventola o dal disco rigido di un computer potrebbe cambiare in risposta al carico di lavoro, che un microfono sensibile potrebbe captare e analizzare per determinare che tipo di operazioni vengono eseguite.

In scenari reali, gli attacchi side-channel hanno effettivamente estratto chiavi di crittografia e compromesso comunicazioni sicure. Uno dei primi casi registrati di un simile attacco si è verificato negli anni '60, quando l'agenzia di intelligence britannica MI5 ha affrontato la sfida di decifrare comunicazioni crittografate dall'ambasciata egiziana a Londra. I loro sforzi di decifrazione dei codici sono stati inizialmente ostacolati dalle limitazioni computazionali dell'epoca, fino a quando un'ingegnosa osservazione dell'agente MI5 Peter Wright ha alterato il corso dell'operazione.

L'agente dell'MI5 Peter Wright propose di usare un microfono per catturare le sottili firme acustiche emesse dalla macchina di cifratura del rotore dell'ambasciata durante la crittografia (Burnet e Thomas 1989). I distinti clic meccanici dei rotori mentre gli operatori li configuravano quotidianamente facevano trapelare informazioni critiche sulle impostazioni iniziali. Questo semplice "canale laterale" del suono ha permesso all'MI5 di ridurre drasticamente la complessità della decifrazione dei messaggi. Questo primo attacco di "perdita" acustica evidenzia che gli attacchi a canale laterale non sono semplicemente una novità dell'era digitale, ma una continuazione di antichi principi di crittoanalisi. L'idea che dove c'è un segnale, c'è un'opportunità di intercettazione rimane fondamentale. Dai clic meccanici alle fluttuazioni elettriche e oltre, i canali laterali consentono agli avversari di estrarre segreti indirettamente attraverso un'attenta analisi del segnale.

Oggi, la crittoanalisi acustica si è evoluta in attacchi come l'intercettazione della tastiera (Asonov e Agrawal 2004). I canali laterali elettrici spaziano dall'analisi della potenza su hardware crittografico (Gnad, Oboril, e Tahoori 2017) alle fluttuazioni di tensione (M. Zhao e Suh 2018) su acceleratori di machine learning. Anche tempistiche, emissioni elettromagnetiche e persino impronte di calore possono essere sfruttate. Nuovi e inaspettati canali laterali emergono spesso man mano che l'informatica diventa più interconnessa e miniaturizzata.

Proprio come la "perdita" acustica analogica dell'MI5 ha trasformato la loro decifrazione dei codici, i moderni attacchi ai canali laterali aggirano i confini tradizionali della difesa informatica. Comprendere lo spirito creativo e la persistenza storica degli exploit dei canali laterali è una conoscenza fondamentale per sviluppatori e difensori che cercano di proteggere in modo completo i moderni sistemi di apprendimento automatico dalle minacce digitali e fisiche.

14.5.5 Interfacce con Perdite

Le interfacce "leaky" nei sistemi embedded sono spesso backdoor trascurate che possono trasformarsi in significative vulnerabilità di sicurezza. Sebbene progettate per scopi legittimi come comunicazione, manutenzione o debug, queste interfacce possono inavvertitamente fornire agli aggressori una finestra attraverso la quale estrarre informazioni sensibili o iniettare dati dannosi.

Un'interfaccia diventa "leaky" quando espone più informazioni del dovuto, spesso a causa della mancanza di rigorosi controlli di accesso o di una schermatura inadeguata dei dati trasmessi. Ecco alcuni esempi concreti di problemi di interfaccia leaky che causano problemi di sicurezza in dispositivi IoT ed embedded:

- **Baby Monitor:** Molti baby monitor abilitati al WiFi hanno interfacce non protette per l'accesso remoto. Ciò ha consentito agli aggressori di ottenere feed audio e video in tempo reale dalle case delle persone, rappresentando una grave violazione della privacy.
- **Pacemaker:** Sono state scoperte vulnerabilità dell'interfaccia in alcuni pacemaker che potrebbero consentire agli aggressori di manipolare le funzioni cardiache se sfruttate. Ciò presenta uno scenario potenzialmente letale.
- **Lampadine Smart:** Un ricercatore ha scoperto di poter accedere a dati non crittografati da lampadine intelligenti tramite un'interfaccia di debug, comprese le credenziali WiFi, consentendogli di accedere alla rete connessa (Greengard 2015).
- **Auto Smart:** Se non protetta, la porta di diagnostica OBD-II ha dimostrato di fornire un vettore di attacco ai sistemi automobilistici. Gli aggressori potrebbero usarlo per controllare i freni e altri componenti (C. Miller e Valasek 2015).

Sebbene quanto sopra non sia direttamente collegato al ML, si consideri l'esempio di un sistema di casa intelligente con un componente ML embedded che controlla la sicurezza domestica in base a pattern di comportamento che apprende nel tempo. Il sistema include un'interfaccia di manutenzione accessibile tramite la rete locale per aggiornamenti software e controlli di sistema. Se questa interfaccia non richiede un'autenticazione forte o i dati trasmessi tramite essa non sono crittografati, un aggressore sulla stessa rete potrebbe ottenere l'accesso. Potrebbero quindi intercettare le routine quotidiane del proprietario di casa o riprogrammare le impostazioni di sicurezza manipolando il firmware.

Tali "fughe" rappresentano un problema di privacy e un potenziale punto di ingresso per exploit più dannosi. L'esposizione di dati di training, parametri del modello o output ML da una "fuga" potrebbe aiutare gli avversari a costruire esempi avversari o a sottoporre a reverse engineering i modelli. L'accesso tramite un'interfaccia con "perdite" potrebbe anche essere

utilizzato per modificare il firmware di un dispositivo embedded, caricandolo con codice dannoso che potrebbe spegnerlo, intercettare dati o utilizzarlo in attacchi botnet.
Per mitigare questi rischi, è necessario un approccio multi-strato, che comprenda controlli tecnici quali autenticazione, crittografia, rilevamento delle anomalie, policy e processi come inventari di interfaccia, controlli di accesso, auditing e pratiche di sviluppo sicure. Disattivare le interfacce non necessarie e compartimentare i rischi tramite un modello zero-trust fornisce una protezione aggiuntiva.
Come progettisti di sistemi ML embedded, dovremmo valutare le interfacce nelle prime fasi dello sviluppo e monitorarle continuamente dopo l'implementazione come parte di un ciclo di vita della sicurezza end-to-end. Comprendere e proteggere le interfacce è fondamentale per garantire la sicurezza complessiva del ML embedded.

14.5.6 Hardware Contraffatto

I sistemi ML sono affidabili solo quanto l'hardware sottostante. In un'epoca in cui i componenti hardware sono beni di consumo globali, l'aumento di hardware contraffatti o clonati rappresenta una sfida significativa. L'hardware contraffatto comprende tutti i componenti che sono riproduzioni non autorizzate di parti originali. I componenti contraffatti si infiltrano nei sistemi ML attraverso complesse catene di fornitura che si estendono oltre i confini e coinvolgono numerose fasi dalla produzione alla consegna.
Anche una sola mancanza di integrità nella catena di fornitura può comportare l'inserimento di parti contraffatte, progettate per imitare fedelmente le funzioni e l'aspetto dell'hardware originale. Ad esempio, un sistema di riconoscimento facciale per il controllo degli accessi ad alta sicurezza potrebbe essere compromesso se dotato di processori contraffatti. Questi processori potrebbero non riuscire a elaborare e verificare accuratamente i dati biometrici, consentendo potenzialmente a persone non autorizzate di accedere ad aree riservate.
La sfida con l'hardware contraffatto è multiforme. Compromette la qualità e l'affidabilità dei sistemi ML, poiché questi componenti potrebbero degradarsi più rapidamente o funzionare in modo imprevedibile a causa di una produzione scadente. Anche i rischi per la sicurezza sono profondi; l'hardware contraffatto può contenere vulnerabilità pronte per essere sfruttate da malintenzionati. Ad esempio, un router di rete clonato in un data center ML potrebbe includere una backdoor nascosta, consentendo l'intercettazione dei dati o l'intrusione nella rete senza essere rilevati.
Inoltre, l'hardware contraffatto comporta rischi legali e di conformità. Le aziende che utilizzano inavvertitamente parti contraffatte nei loro sistemi ML possono affrontare gravi ripercussioni legali, tra cui multe e sanzioni per il mancato rispetto delle normative e degli standard del settore. Ciò è particolarmente vero per i settori in cui è obbligatoria la conformità a specifiche normative sulla sicurezza e sulla privacy, come l'assistenza sanitaria e la finanza.
Le pressioni economiche per ridurre i costi aggravano il problema dell'hardware contraffatto e costringono le aziende ad approvvigionarsi da fornitori a basso costo, privi di rigorosi processi di verifica. Questa economia può introdurre inavvertitamente parti contraffatte in sistemi altrimenti sicuri. Inoltre, rilevare queste contraffazioni è intrinsecamente complicato poiché vengono create per passare per componenti originali, il che spesso richiede attrezzature e competenze sofisticate per essere identificate.
Nel campo dell'apprendimento automatico, dove decisioni in tempo reale e calcoli complessi sono la norma, le implicazioni di un guasto hardware possono essere scomode e potenzialmente pericolose. È fondamentale che le parti interessate siano pienamente consapevoli di questi rischi. Le sfide poste dall'hardware contraffatto richiedono una comprensione completa delle attuali minacce all'integrità del sistema di apprendimento

automatico. Ciò sottolinea la necessità di una gestione proattiva e informata del ciclo di vita dell'hardware all'interno di questi sistemi avanzati.

14.5.7 Rischi della Catena di Fornitura

La minaccia dell'hardware contraffatto è strettamente legata alle vulnerabilità più ampie della supply chain [catena di fornitura]. Le supply chain globalizzate e interconnesse creano molteplici opportunità per componenti compromessi di infiltrarsi nel ciclo di vita di un prodotto. Le supply chain coinvolgono numerose entità, dalla progettazione alla produzione, all'assemblaggio, alla distribuzione e all'integrazione. Una mancanza di trasparenza e supervisione di ogni partner rende difficile la verifica dell'integrità a ogni passaggio. Le lacune in qualsiasi punto della catena possono consentire l'inserimento di parti contraffatte.

Ad esempio, un produttore su contratto potrebbe ricevere e includere inconsapevolmente rifiuti elettronici riciclati contenenti contraffazioni pericolose. Un distributore inaffidabile potrebbe introdurre di nascosto componenti clonati. Le minacce interne a qualsiasi fornitore potrebbero deliberatamente mescolare contraffazioni in spedizioni legittime.

Una volta che le contraffazioni entrano nel flusso di fornitura, passano rapidamente attraverso più mani prima di finire nei sistemi ML in cui il rilevamento è difficile. Le contraffazioni avanzate come parti ricondizionate o cloni con esterni riconfezionati possono mascherarsi da componenti autentici, superando l'ispezione visiva.

Per identificare i falsi, spesso è richiesta una profilazione tecnica completa tramite micrografia, screening a raggi X, analisi forense dei componenti e test funzionali. Tuttavia, un'analisi così costosa non è pratica per gli acquisti su larga scala.

Strategie come audit della supply chain, screening dei fornitori, convalida della provenienza dei componenti e aggiunta di protezioni antimanomissione possono aiutare a mitigare i rischi. Tuttavia, date le sfide globali alla sicurezza della supply chain, un approccio zero-trust è prudente. Progettare sistemi ML per utilizzare controlli ridondanti, fail-safe e monitoraggio continuo del runtime fornisce resilienza contro i compromessi dei componenti.

Una rigorosa convalida delle sorgenti hardware abbinata ad architetture di sistema fault-tolerant offre la difesa più solida contro i rischi pervasivi di supply chain globali contorte e opache.

14.5.8 Caso di Studio

Nel 2018, Bloomberg Businessweek ha pubblicato una storia allarmante che ha attirato molta attenzione nel mondo della tecnologia. L'articolo sosteneva che Supermicro aveva segretamente impiantato minuscoli chip spia nell'hardware del server. I giornalisti hanno affermato che gli hacker statali cinesi che lavoravano con Supermicro potevano infilare questi minuscoli chip nelle schede madri durante la produzione. I minuscoli chip avrebbero presumibilmente dato agli hacker un accesso backdoor ai server utilizzati da oltre 30 grandi aziende, tra cui Apple e Amazon.

Se fosse vero, ciò consentirebbe agli hacker di spiare dati privati o persino manomettere i sistemi. Tuttavia, dopo aver indagato, Apple e Amazon non hanno trovato prove dell'esistenza di tale hardware Supermicro hackerato. Altri esperti hanno messo in dubbio l'accuratezza dell'articolo di Bloomberg.

Se la storia sia del tutto accurata o meno non è una nostra preoccupazione da un punto di vista pedagogico. Tuttavia, questo incidente ha attirato l'attenzione sui rischi delle catene di fornitura globali per l'hardware prodotto principalmente in Cina. Quando le aziende esternalizzano e acquistano componenti hardware da fornitori in tutto il mondo, è necessario che vi sia maggiore visibilità nel processo. In questa complessa pipeline globale, si teme che hardware contraffatti o manomessi possano essere introdotti da qualche parte lungo il percorso senza che le aziende tecnologiche se ne accorgano. Le aziende che si affidano

troppo a singoli produttori o distributori creano rischi. Ad esempio, a causa dell'eccessiva dipendenza da TSMC per la produzione di semiconduttori, gli Stati Uniti hanno investito 50 miliardi di dollari nel CHIPS Act.
Man mano che l'apprendimento automatico si sposta in sistemi più critici, è fondamentale verificare l'integrità dell'hardware dalla progettazione alla produzione e alla consegna. La backdoor Supermicro segnalata ha dimostrato che per la sicurezza dell'apprendimento automatico non possiamo dare per scontate le catene di fornitura e la produzione globali. Dobbiamo ispezionare e convalidare l'hardware a ogni collegamento della catena.

14.6 Sicurezza Hardware del ML Embedded

14.6.1 Trusted Execution Environments

14.6.1.1 Informazioni su TEE

Un Trusted Execution Environment (TEE) è un'area protetta all'interno di un processore host che garantisce l'esecuzione sicura del codice e la protezione dei dati sensibili. Isolando le attività critiche dal sistema operativo, i TEE resistono agli attacchi software e hardware, fornendo un ambiente protetto per la gestione di calcoli sensibili.

14.6.1.2 Vantaggi

I TEE sono particolarmente preziosi in scenari in cui devono essere elaborati dati sensibili o in cui l'integrità delle operazioni di un sistema è critica. Nel contesto dell'hardware ML, i TEE assicurano che gli algoritmi e i dati ML siano protetti da manomissioni e "perdite". Ciò è essenziale perché i modelli ML elaborano spesso informazioni private, segreti commerciali o dati che potrebbero essere sfruttati se esposti.
Ad esempio, un TEE può proteggere i parametri del modello ML dall'estrazione da parte di software dannosi sullo stesso dispositivo. Questa protezione è fondamentale per la privacy e il mantenimento dell'integrità del sistema ML, assicurando che i modelli funzionino come previsto e non forniscano output distorti a causa di parametri manipolati. Secure Enclave di Apple, presente in iPhone e iPad, è una forma di TEE che fornisce un ambiente isolato per proteggere i dati sensibili degli utenti e le operazioni crittografiche.
I Trusted Execution Environment (TEE) sono essenziali per i settori che richiedono elevati livelli di sicurezza, tra cui telecomunicazioni, finanza, sanità e automotive. I TEE proteggono l'integrità delle reti 5G nelle telecomunicazioni e supportano applicazioni critiche. Nella finanza, proteggono i pagamenti mobili e i processi di autenticazione. L'assistenza sanitaria si affida ai TEE per salvaguardare i dati sensibili dei pazienti, mentre il settore automobilistico dipende da loro per la sicurezza e l'affidabilità dei sistemi autonomi. In tutti i settori, i TEE garantiscono la riservatezza e l'integrità dei dati e delle operazioni.
Nei sistemi ML, i TEE possono:

- Eseguire in modo sicuro l'addestramento e l'inferenza del modello, assicurando che i risultati del calcolo rimangano riservati.
- Proteggere la riservatezza dei dati di input, come le informazioni biometriche, utilizzati per l'identificazione personale o per attività di classificazione sensibili.
- Proteggere i modelli ML impedendo il reverse engineering, che può proteggere le informazioni proprietarie e mantenere un vantaggio competitivo.
- Abilitare aggiornamenti sicuri ai modelli ML, assicurando che gli aggiornamenti provengano da una fonte attendibile e non siano stati manomessi durante il transito.
- Rafforzare la sicurezza della rete salvaguardando la trasmissione dei dati tra componenti ML distribuiti tramite crittografia ed elaborazione sicura in-TEE.

L'importanza dei TEE nella sicurezza hardware ML deriva dalla loro capacità di proteggere da minacce esterne e interne, tra cui le seguenti:

- **Software Dannoso:** I TEE possono impedire al malware ad alto privilegio di accedere alle aree sensibili del sistema ML.
- **Manomissione Fisica:** Integrandosi con le misure di sicurezza hardware, i TEE possono proteggere dalla manomissione fisica che tenta di aggirare la sicurezza del software.
- **Attacchi Side-Channel:** Sebbene non siano impenetrabili, i TEE possono mitigare specifici attacchi side-channel controllando l'accesso a operazioni sensibili e modelli di dati.
- **Minacce di Rete:** I TEE migliorano la sicurezza della rete salvaguardando la trasmissione dei dati tra componenti ML distribuiti tramite crittografia ed elaborazione in-TEE sicura. Ciò impedisce efficacemente gli attacchi "man-in-the-middle" e garantisce che i dati vengano trasmessi tramite canali attendibili.

14.6.1.3 Meccanica

I fondamenti dei TEE contengono quattro parti principali:

- **Esecuzione Isolata:** Il codice all'interno di un TEE viene eseguito in un ambiente separato dal sistema operativo host del dispositivo host. Questo isolamento protegge il codice dall'accesso non autorizzato da parte di altre applicazioni.
- **Archiviazione Sicura:** I TEE possono archiviare in modo sicuro chiavi crittografiche, token di autenticazione e dati sensibili, impedendo alle applicazioni normali di accedervi al di fuori del TEE.
- **Protezione dell'Integrità:** I TEE possono verificare l'integrità del codice e dei dati, assicurando che non siano stati alterati prima dell'esecuzione o durante l'archiviazione.
- **Crittografia dei Dati:** I dati gestiti all'interno di un TEE possono essere crittografati, rendendoli illeggibili per entità senza le chiavi appropriate, che sono anch'esse gestite all'interno del TEE.

Ecco alcuni esempi di TEE che forniscono sicurezza basata su hardware per applicazioni sensibili:

- **ARMTrustZone:** Questa tecnologia crea ambienti di esecuzione sicuri e normali isolati tramite controlli hardware e implementati in molti chipset mobili. mobile chipsets.
- **IntelSGX:** Le estensioni Software Guard di Intel forniscono un'enclave per l'esecuzione del codice che protegge da varie minacce basate sul software, prendendo di mira in modo specifico le vulnerabilità del livello del sistema operativo. Vengono utilizzate per salvaguardare i carichi di lavoro nel cloud.
- **Qualcomm Secure Execution Environment:** Un sandbox hardware su chipset Qualcomm per app di pagamento e autenticazione mobili.
- **Apple SecureEnclave:** Un TEE per la gestione dei dati biometrici e delle chiavi crittografiche su iPhone e iPad, che facilita i pagamenti mobili sicuri.

Figura 14.7 è un diagramma che mostra un'enclave sicura isolata dal processore host per fornire un ulteriore livello di sicurezza. L'enclave sicura ha una ROM di avvio per stabilire una "root" hardware di attendibilità, un motore AES per operazioni crittografiche efficienti e sicure e memoria protetta. Ha anche un meccanismo per memorizzare le informazioni in modo sicuro su un archivio collegato separato da quello flash NAND utilizzato dal processore applicativo e dal sistema operativo. Questo design mantiene al sicuro i dati sensibili degli utenti anche quando il kernel dell'Application Processor viene compromesso.

Figura 14.7: Enclave sicura System-on-chip. Fonte: Apple.

14.6.1.4 Compromessi

Sebbene i "Trusted Execution Environment" offrano notevoli vantaggi in termini di sicurezza, la loro implementazione comporta dei compromessi. Diversi fattori influenzano se un sistema include un TEE:

Costo: L'implementazione dei TEE comporta costi aggiuntivi. Ci sono costi diretti per l'hardware e costi indiretti associati allo sviluppo e alla manutenzione di software sicuro per i TEE. Questi costi potrebbero essere giustificabili solo per alcuni dispositivi, in particolare prodotti a basso margine.

Complessità: I TEE aggiungono complessità alla progettazione e allo sviluppo del sistema. L'integrazione di un TEE con sistemi esistenti richiede una sostanziale ri-progettazione dello

stack hardware e software, che può rappresentare un ostacolo, in particolare per i sistemi legacy.

Performance Overhead: I TEE possono introdurre un overhead di prestazioni dovuto ai passaggi aggiuntivi coinvolti nella crittografia e nella verifica dei dati, che potrebbero rallentare le applicazioni sensibili al tempo.

Sfide di Sviluppo: Lo sviluppo per i TEE richiede conoscenze specialistiche e spesso deve rispettare rigidi protocolli di sviluppo. Ciò può estendere i tempi di sviluppo e complicare i processi di debug e test.

Scalabilità e Flessibilità: I TEE, a causa della loro natura protetta, possono imporre limitazioni di scalabilità e flessibilità. L'aggiornamento dei componenti protetti o il ridimensionamento del sistema per più utenti o dati può essere più impegnativo quando tutto deve passare attraverso un ambiente protetto e chiuso.

Consumo Energetico: L'elaborazione aumentata richiesta per la crittografia, la decrittografia e i controlli di integrità possono portare a un maggiore consumo energetico, una preoccupazione significativa per i dispositivi alimentati a batteria.

Domanda del Mercato: Non tutti i mercati o le applicazioni richiedono il livello di sicurezza fornito dai TEE. Per molte applicazioni consumer, il rischio percepito potrebbe essere abbastanza basso da indurre i produttori a scegliere di non includere TEE nei loro progetti.

Certificazione e Garanzia di Sicurezza: I sistemi con TEE potrebbero aver bisogno di rigorose certificazioni di sicurezza con enti come Common Criteria (CC) o European Union Agency for Cybersecurity (ENISA), che possono essere lunghe e costose. Alcune organizzazioni potrebbero scegliere di astenersi dall'implementare TEE per evitare questi ostacoli.

Dispositivi con risorse limitate: I dispositivi con potenza di elaborazione, memoria o archiviazione limitate potrebbero supportare solo TEE senza compromettere la loro funzionalità primaria.

14.6.2 Avvio Sicuro

14.6.2.1 Informazioni

Secure Boot è uno standard di sicurezza fondamentale che garantisce che un dispositivo si avvii solo tramite software attendibile dal "Original Equipment Manufacturer (OEM)" [produttore di apparecchiature originali]. Durante l'avvio, il firmware controlla la firma digitale di ogni componente software di avvio, inclusi bootloader, kernel e sistema operativo di base. Questo processo verifica che il software non sia stato alterato o manomesso. Se una firma non supera la verifica, il processo di avvio viene interrotto per impedire l'esecuzione di codice non autorizzato che potrebbe compromettere l'integrità della sicurezza del sistema.

14.6.2.2 Vantaggi

L'integrità di un sistema di apprendimento automatico (ML) embedded è fondamentale dal momento in cui viene acceso. Qualsiasi compromissione nel processo di avvio può portare all'esecuzione di software dannoso prima che il sistema operativo e le applicazioni ML inizino, con conseguenti operazioni ML manipolate, accesso ai dati non autorizzato o riutilizzo del dispositivo per attività dannose come botnet o crypto-mining.
Secure Boot offre protezioni vitali per l'hardware ML embedded tramite i seguenti meccanismi critici:

- **Protezione dei Dati ML:** Garantire che i dati utilizzati dai modelli ML, che possono includere informazioni private o sensibili, non siano esposti a manomissioni o furti durante il processo di boot [avvio].
- **Protezione dell'Integrità del Modello:** Mantenere l'integrità dei modelli ML è fondamentale, poiché la loro manomissione potrebbe portare a risultati errati o dannosi.
- **Aggiornamenti Sicuri del Modello:** Abilitare aggiornamenti sicuri per modelli e algoritmi ML, assicurando che gli aggiornamenti siano autenticati e non siano stati alterati.

14.6.2.3 Meccanica

Secure Boot funziona con i TEE per migliorare ulteriormente la sicurezza del sistema. Figura 14.8 illustra un diagramma di flusso di un sistema embedded affidabile. Nella fase di validazione iniziale, Secure Boot verifica che il codice in esecuzione nel TEE sia la versione corretta e non manomessa autorizzata dal produttore del dispositivo. Controllando le firme digitali del firmware e di altri componenti critici del sistema, Secure Boot impedisce modifiche non autorizzate che potrebbero compromettere le capacità di sicurezza del TEE. Ciò stabilisce una base di fiducia su cui il TEE può eseguire in modo sicuro operazioni sensibili come la gestione delle chiavi crittografiche e l'elaborazione sicura dei dati. Applicando questi livelli di sicurezza, Secure Boot consente operazioni dei dispositivi sicure e resilienti anche negli ambienti con risorse più limitate.

Figura 14.8: Flusso di Secure Boot. Fonte: R. V. e A. (2018).

14.6.2.4 Caso di Studio: Face ID di Apple

Un esempio concreto dell'applicazione di Secure Boot può essere osservato nella tecnologia Face ID di Apple, che utilizza algoritmi di apprendimento automatico avanzati per abilitare il riconoscimento facciale su iPhone e iPad. Face ID si basa su una sofisticata integrazione di sensori e software per mappare con precisione la geometria del volto di un utente. Affinché Face ID funzioni in modo sicuro e protegga i dati biometrici degli utenti, le operazioni del dispositivo devono essere affidabili fin dall'inizializzazione. È qui che Secure Boot svolge un ruolo fondamentale. Di seguito viene descritto come funziona Secure Boot insieme a Face ID:

Verifica Iniziale: All'avvio di un iPhone, il processo Secure Boot inizia all'interno di Secure Enclave, un coprocessore specializzato progettato per aggiungere un ulteriore livello di sicurezza. Secure Enclave gestisce i dati biometrici, come le impronte digitali per Touch ID e i dati di riconoscimento facciale per Face ID. Durante il processo di avvio, il sistema verifica rigorosamente che Apple abbia firmato digitalmente il firmware di Secure Enclave, garantendone l'autenticità. Questa fase di verifica assicura che il firmware utilizzato per elaborare i dati biometrici rimanga sicuro e senza essere compromesso.

Controlli di sicurezza continui: Dopo l'inizializzazione e la convalida del sistema da parte di Secure Boot, Secure Enclave comunica con il processore centrale del dispositivo per mantenere una catena di avvio sicura. Durante questo processo, le firme digitali del kernel iOS e di altri componenti di avvio critici vengono meticolosamente verificate per garantirne l'integrità prima di procedere. Questo modello di "catena di fiducia" impedisce efficacemente modifiche non autorizzate al bootloader e al sistema operativo, salvaguardando la sicurezza complessiva del dispositivo.

Elaborazione dei Dati del Viso: Una volta completata la sequenza di avvio sicura, Secure Enclave interagisce in modo sicuro con gli algoritmi di apprendimento automatico che alimentano Face ID. Il riconoscimento facciale prevede la proiezione e l'analisi di oltre 30.000 punti invisibili per creare una mappa di profondità del volto dell'utente e un'immagine a infrarossi. Questi dati vengono convertiti in una rappresentazione matematica e confrontati in modo sicuro con i dati del volto registrati e archiviati in Secure Enclave.

Secure Enclave e Protezione dei Dati: Secure Enclave è progettato con precisione per proteggere i dati sensibili e gestire le operazioni crittografiche che salvaguardano tali dati. Anche in caso di kernel del sistema operativo compromesso, i dati del volto elaborati tramite Face ID rimangono inaccessibili ad applicazioni non autorizzate o ad aggressori esterni. È importante sottolineare che i dati di Face ID non vengono mai trasmessi dal dispositivo e non vengono archiviati su iCloud o altri server esterni.

Aggiornamenti Firmware: Apple rilascia frequentemente aggiornamenti per risolvere le vulnerabilità della sicurezza e migliorare la funzionalità del sistema. Secure Boot garantisce che tutti gli aggiornamenti del firmware siano autenticati, consentendo l'installazione solo di quelli firmati da Apple. Questo processo aiuta a preservare l'integrità e la sicurezza del sistema di Face ID nel tempo.
Integrando Secure Boot con hardware dedicato come Secure Enclave, Apple offre solide garanzie di sicurezza per operazioni critiche come il riconoscimento facciale.

14.6.2.5 Sfide

Nonostante i suoi vantaggi, l'implementazione di Secure Boot presenta diverse sfide, in particolare in distribuzioni complesse e su larga scala: **Complessità della Gestione delle Chiavi:** Generare, archiviare, distribuire, ruotare e revocare chiavi crittografiche in modo dimostrabilmente sicuro è particolarmente impegnativo ma fondamentale per mantenere la

catena di fiducia. Qualsiasi compromissione delle chiavi paralizza le protezioni. Le grandi aziende che gestiscono moltitudini di chiavi di dispositivi affrontano particolari sfide di scala.

Sovraccarico di Prestazioni: Il controllo delle firme crittografiche durante il Boot può aggiungere 50-100ms o più per componente verificato. Questo ritardo può essere proibitivo per applicazioni sensibili al tempo o con risorse limitate. Tuttavia, gli impatti sulle prestazioni possono essere ridotti tramite parallelizzazione e accelerazione hardware.

Signing Burden: [Onere della firma] Gli sviluppatori devono garantire diligentemente che tutti i componenti software coinvolti nel processo di avvio, ovvero bootloader, firmware, kernel del sistema operativo, driver, applicazioni, ecc., siano firmati correttamente da chiavi attendibili. L'accettazione della firma del codice di terze parti rimane un problema.

Verifica Crittografica: Gli algoritmi e i protocolli sicuri devono convalidare la legittimità di chiavi e firme, evitare manomissioni o bypass e supportare la revoca. L'accettazione di chiavi dubbie mina la fiducia.

Vincoli di Personalizzazione: Le architetture Secure Boot bloccate dal fornitore limitano il controllo dell'utente e l'aggiornabilità. I bootloader open source come u-boot e coreboot abilitano la sicurezza supportando al contempo la personalizzazione.

Standard Scalabili: Standard emergenti come Device Identifier Composition Engine (DICE) e IDevID promettono di fornire e gestire in modo sicuro identità e chiavi dei dispositivi su larga scala in tutti gli ecosistemi.
L'adozione di Secure Boot richiede di seguire le best practice di sicurezza relative alla gestione delle chiavi, alla convalida della crittografia, agli aggiornamenti firmati e al controllo degli accessi. Secure Boot fornisce una solida base per creare integrità e affidabilità dei dispositivi se implementato con cura.

14.6.3 Moduli di Sicurezza Hardware

14.6.3.1 HSM

Un "Hardware Security Module (HSM)" è un dispositivo fisico che gestisce le chiavi digitali per un'autenticazione avanzata e fornisce l'elaborazione crittografica. Questi moduli sono progettati per essere resistenti alle manomissioni e fornire un ambiente sicuro per l'esecuzione di operazioni crittografiche. Gli HSM possono essere dispositivi standalone, schede plug-in o circuiti integrati su un altro dispositivo.
Gli HSM sono fondamentali per varie applicazioni sensibili alla sicurezza perché offrono un'enclave rafforzata e sicura per l'archiviazione delle chiavi crittografiche e l'esecuzione di funzioni crittografiche. Sono particolarmente importanti per garantire la sicurezza delle transazioni, le verifiche dell'identità e la crittografia dei dati.

14.6.3.2 Vantaggi

Gli HSM forniscono diverse funzionalità utili per la sicurezza dei sistemi ML:

Protezione dei Dati Sensibili: Nelle applicazioni di apprendimento automatico, i modelli spesso elaborano dati sensibili che possono essere proprietari o personali. Gli HSM proteggono le chiavi di crittografia utilizzate per proteggere questi dati, sia a riposo che in transito, dall'esposizione o dal furto.

Garanzia dell'Integrità del Modello: L'integrità dei modelli ML è fondamentale per il loro funzionamento affidabile. Gli HSM possono gestire in modo sicuro i processi di firma e

verifica per software e firmware ML, assicurando che parti non autorizzate non abbiano alterato i modelli.

Addestramento e Aggiornamenti Sicuri del Modello: L'addestramento e l'aggiornamento dei modelli ML comportano l'elaborazione di dati potenzialmente sensibili. Gli HSM garantiscono che questi processi vengano condotti all'interno di un confine crittografico sicuro, proteggendo dall'esposizione dei dati di addestramento e dagli aggiornamenti non autorizzati del modello.

14.6.3.3 Compromessi

Gli HSM comportano diversi compromessi per l'ML embedded. Questi compromessi sono simili ai TEE, ma per completezza, li discuteremo anche qui attraverso la lente dell'HSM.

Costo: Gli HSM sono dispositivi specializzati che possono essere costosi da procurare e implementare, aumentando il costo complessivo di un progetto ML. Questo può essere un fattore significativo per i sistemi embedded, dove i vincoli di costo sono spesso più rigidi.

Sovraccarico di Prestazioni: Sebbene sicure, le operazioni crittografiche eseguite dagli HSM possono introdurre latenza. Qualsiasi ritardo aggiunto può essere critico nelle applicazioni ML embedded ad alte prestazioni in cui l'inferenza deve avvenire in tempo reale, come nei veicoli autonomi o nei dispositivi di traduzione.

Spazio Fisico: I sistemi embedded sono spesso limitati dallo spazio fisico e l'aggiunta di un HSM può essere difficile in ambienti con vincoli rigidi. Ciò è particolarmente vero per l'elettronica di consumo e la tecnologia indossabile, dove le dimensioni e il fattore di forma sono considerazioni chiave.

Consumo Energetico: Gli HSM richiedono energia per funzionare, il che può rappresentare uno svantaggio per i dispositivi a batteria con una lunga durata della batteria. L'elaborazione sicura e le operazioni crittografiche possono scaricare la batteria più velocemente, un compromesso significativo per le applicazioni ML embedded mobili o remote.

Complessità nell'Integrazione: L'integrazione degli HSM nei sistemi hardware esistenti aggiunge complessità. Spesso sono necessarie conoscenze specialistiche per gestire la comunicazione sicura tra l'HSM e il processore del sistema e sviluppare software in grado di interfacciarsi con l'HSM.

Scalabilità: Il ridimensionamento di una soluzione ML che utilizza gli HSM può essere impegnativo. Gestire una flotta di HSM e garantire l'uniformità nelle pratiche di sicurezza tra i dispositivi può diventare complesso e costoso quando aumentano le dimensioni della distribuzione, soprattutto quando si ha a che fare con sistemi embedded in cui la comunicazione è costosa.

Complessità Operativa: Gli HSM possono rendere più complesso l'aggiornamento del firmware e dei modelli ML. Ogni aggiornamento deve essere firmato e possibilmente crittografato, il che aggiunge passaggi al processo di aggiornamento e potrebbe richiedere meccanismi sicuri per la gestione delle chiavi e la distribuzione degli aggiornamenti.

Sviluppo e Manutenzione: La natura sicura degli HSM implica che solo personale limitato abbia accesso all'HSM per scopi di sviluppo e manutenzione. Ciò può rallentare il processo di sviluppo e rendere più difficile la manutenzione di routine.

Certificazione e Conformità: Garantire che un HSM soddisfi specifici standard di settore e requisiti di conformità può aumentare i tempi e i costi di sviluppo. Ciò potrebbe comportare l'esecuzione di rigorosi processi di certificazione e audit.

14.6.4 Physical Unclonable Functions (PUF)

14.6.4.1 Informazioni

Le “Physical Unclonable Function (PUF)” [funzioni fisiche non clonabili] forniscono un mezzo intrinseco all’hardware per la generazione di chiavi crittografiche e l’autenticazione del dispositivo sfruttando la variabilità di produzione intrinseca nei componenti semiconduttori. Durante la fabbricazione, fattori fisici casuali come variazioni di drogaggio, ruvidità del bordo della linea e spessore dielettrico determinano differenze microscopiche tra i semiconduttori, anche quando prodotti dalle stesse maschere. Questi creano variazioni di temporizzazione e potenza rilevabili che agiscono come una “impronta digitale” unica per ogni chip. Le PUF sfruttano questo fenomeno incorporando circuiti integrati per amplificare piccole differenze di temporizzazione o potenza in uscite digitali misurabili.
Quando stimolato con uno stimolo in input, il circuito PUF produce una risposta di output basata sulle caratteristiche fisiche intrinseche del dispositivo. A causa della loro unicità fisica, lo stesso stimolo produrrà una risposta diversa su altri dispositivi. Questo meccanismo di stimolo-risposta può essere utilizzato per generare chiavi in modo sicuro e identificatori legati all’hardware specifico, eseguire l’autenticazione del dispositivo o archiviare in modo sicuro i segreti. Ad esempio, una chiave derivata da un PUF funzionerà solo su quel dispositivo e non potrà essere clonata o estratta nemmeno con accesso fisico o reverse engineering completo (Gao, Al-Sarawi, e Abbott 2020).

14.6.4.2 Vantaggi

La generazione di chiavi PUF evita l’archiviazione esterna delle chiavi, che rischia di essere esposta. Fornisce inoltre una base per altre primitive di sicurezza hardware come Secure Boot. Le sfide di implementazione includono la gestione di affidabilità ed entropia variabili tra diverse PUF, sensibilità alle condizioni ambientali e suscettibilità agli attacchi di modellazione di apprendimento automatico. Se progettate con cura, le PUF consentono applicazioni promettenti nella protezione IP, nel trusted computing e nell’anticontraffazione.

14.6.4.3 Utilità

I modelli di apprendimento automatico stanno rapidamente diventando una parte fondamentale della funzionalità per molti dispositivi embedded, come smartphone, assistenti domestici intelligenti e droni autonomi. Tuttavia, proteggere l’apprendimento automatico su hardware embedded con risorse limitate può essere difficile. È qui che le funzioni fisiche non clonabili (PUF) risultano particolarmente utili. Diamo un’occhiata ad alcuni esempi di come le PUF possono essere utili.
Le PUF forniscono un modo per generare impronte digitali e chiavi crittografiche univoche legate alle caratteristiche fisiche di ciascun chip sul dispositivo. Facciamo un esempio. Abbiamo un drone con telecamera intelligente che usa ML embedded per tracciare gli oggetti. Un PUF integrato nel processore del drone potrebbe creare una chiave specifica del dispositivo per crittografare il modello ML prima di caricarlo sul drone. In questo modo, anche se un aggressore in qualche modo hackerasse il drone e provasse a rubare il modello, non sarebbe in grado di usarlo su un altro dispositivo!
La stessa chiave PUF potrebbe anche creare una filigrana digitale embedded nel modello ML. Se quel modello dovesse mai trapelare e essere pubblicato online da qualcuno che cercasse di piratarlo, la filigrana potrebbe aiutare a dimostrare che proviene dal drone rubato e non dall’aggressore. Inoltre, si immagini che la telecamera del drone si colleghi al cloud per scaricare parte della sua elaborazione ML. Il PUF può autenticare che la telecamera è legittima prima che il cloud esegua l’inferenza su video sensibili. Il cloud potrebbe verificare che il drone non sia stato manomesso fisicamente controllando che le risposte PUF non siano cambiate.

Le PUF consentono tutta questa sicurezza attraverso la casualità intrinseca del loro comportamento di stimolo-risposta e il binding hardware. Senza dover memorizzare le chiavi esternamente, le PUF sono ideali per proteggere l'ML embedded con risorse limitate. Pertanto, offrono un vantaggio unico rispetto ad altri meccanismi.

14.6.4.4 Meccanica

Il principio di funzionamento alla base dei PUF, illustrato in Figura 14.9, comporta la generazione di una coppia "stimolo-risposta", in cui un input specifico (lo stimolo) al circuito PUF determina un output (la risposta) che è determinato dalle proprietà fisiche uniche di quel circuito. Questo processo può essere paragonato a un meccanismo di impronte digitali per dispositivi elettronici. I dispositivi che utilizzano ML per elaborare i dati dei sensori possono utilizzare i PUF per proteggere la comunicazione tra dispositivi e impedire l'esecuzione di modelli ML su hardware contraffatto.

Figura 14.9 illustra una panoramica delle basi dei PUF: a) PUF può essere pensato come un'impronta digitale unica per ogni pezzo di hardware; b) un PUF Ottico è uno speciale token di plastica che viene illuminato, creando un pattern a macchie unico che viene poi registrato; c) in un APUF (Arbiter PUF), i bit di stimolo selezionano percorsi diversi e un giudice decide quale è più veloce, dando una risposta di '1' o '0'; d) in un PUF SRAM, la risposta è determinata dalla mancata corrispondenza nella tensione di soglia dei transistor, dove determinate condizioni portano a una risposta preferita di '1'. Ognuno di questi metodi utilizza caratteristiche specifiche dell'hardware per creare un identificatore univoco.

Figura 14.9: Nozioni di base sui PUF. Fonte: Gao, Al-Sarawi, e Abbott (2020).

14.6.4.5 Sfide

Ci sono alcune sfide con i PUF. La risposta PUF può essere sensibile alle condizioni ambientali, come fluttuazioni di temperatura e tensione, portando a un comportamento incoerente di cui si deve tenere conto nella progettazione. Inoltre, poiché i PUF possono generare molte coppie stimolo-risposta uniche, gestire e garantire la coerenza di queste

coppie per tutta la durata del dispositivo può essere impegnativo. Ultimo ma non meno importante, l'integrazione della tecnologia PUF può aumentare il costo di produzione complessivo di un dispositivo, sebbene possa far risparmiare sui costi di gestione delle chiavi durante il ciclo di vita del dispositivo.

14.7 Problemi di Privacy nella Gestione dei Dati

La gestione sicura ed etica dei dati personali e sensibili è fondamentale poiché l'apprendimento automatico permea dispositivi come smartphone, dispositivi indossabili ed elettrodomestici intelligenti. Per l'hardware medico, la gestione sicura ed etica dei dati è ulteriormente richiesta dalla legge tramite l'Health Insurance Portability and Accountability Act (HIPAA). Questi sistemi ML embedded presentano rischi unici per la privacy, data la loro intima vicinanza alla vita degli utenti.

14.7.1 Tipi di Dati Sensibili

I dispositivi ML embedded come quelli indossabili, assistenti domestici intelligenti e veicoli autonomi elaborano spesso dati altamente personali che richiedono un'attenta gestione per preservare la privacy dell'utente e prevenirne l'uso improprio. Esempi specifici includono referti medici e piani di trattamento elaborati da dispositivi indossabili per la salute, conversazioni private costantemente acquisite da assistenti domestici intelligenti e abitudini di guida dettagliate raccolte da auto connesse. La compromissione di tali dati sensibili può portare a gravi conseguenze come furto di identità, manipolazione emotiva, umiliazione pubblica ed abuso di sorveglianza di massa.

I dati sensibili assumono molte forme: registri strutturati come elenchi di contatti e contenuti non strutturati come flussi audio e video conversazionali. In ambito medico, le "protected health information (PHI)" [informazioni sanitarie protette] vengono raccolte dai medici durante ogni interazione e sono fortemente regolamentate da rigide linee guida HIPAA. Anche al di fuori degli ambienti medici, i dati sensibili possono comunque essere raccolti sotto forma di Personally Identifiable Information (PII) [Informazioni di identificazione personale], definite come "qualsiasi rappresentazione di informazioni che consenta di dedurre ragionevolmente l'identità di un individuo a cui si applicano le informazioni con mezzi diretti o indiretti". Esempi di PII includono indirizzi e-mail, numeri di previdenza sociale e numeri di telefono, tra gli altri campi. Le PII vengono raccolte in ambito medico e in altri contesti (applicazioni finanziarie, ecc.) e sono fortemente regolamentate dalle politiche del Dipartimento del Lavoro.

Anche gli output dei modelli derivati potrebbero far trapelare indirettamente dettagli sugli individui. Oltre ai dati personali, anche algoritmi e set di dati proprietari garantiscono la protezione della riservatezza. Nella sezione Data Engineering, abbiamo trattato diversi argomenti in dettaglio.

Tecniche come la de-identificazione, l'aggregazione, l'anonimizzazione e la federazione possono aiutare a trasformare i dati sensibili in forme meno rischiose mantenendo al contempo l'utilità analitica. Tuttavia, controlli diligenti su accesso, crittografia, auditing, consenso, minimizzazione e pratiche di conformità sono ancora essenziali durante tutto il ciclo di vita dei dati. Regolamenti come GDPR categorizzano diverse classi di dati sensibili e prescrivono responsabilità in merito alla loro gestione etica. Standard come NIST 800-53 forniscono rigorose linee guida per il controllo della sicurezza per la protezione della riservatezza. Con la crescente dipendenza dal ML embedded, comprendere i rischi dei dati sensibili è fondamentale.

14.7.2 Regolamenti Applicabili

Molte applicazioni ML embedded gestiscono dati sensibili degli utenti in base alle normative HIPAA, GDPR e CCPA. Comprendere le protezioni imposte da queste leggi è fondamentale per creare sistemi conformi.

- La norma sulla privacy HIPAA stabilisce che i fornitori di assistenza che svolgono determinate attività regolano la privacy e la sicurezza dei dati medici negli Stati Uniti, con severe sanzioni per le violazioni. Tutti i dispositivi ML embedded correlati alla salute, come dispositivi indossabili diagnostici o robot di assistenza, dovrebbero implementare controlli come audit trail, controlli di accesso e crittografia prescritti da HIPAA.
- Il GDPR impone trasparenza, limiti di conservazione e diritti degli utenti sui dati dei cittadini dell'UE, anche quando elaborati da aziende al di fuori dell'UE. I sistemi per la casa intelligente che catturano conversazioni familiari o pattern di posizione dovrebbero essere conformi al GDPR. I requisiti chiave includono la minimizzazione dei dati, la crittografia e meccanismi per il consenso e la cancellazione.
- Il CCPA, che si applica in California, protegge la privacy dei dati dei consumatori tramite disposizioni quali divulgazioni obbligatorie e diritti di opt-out: i gadget IoT come gli smart speaker e i fitness tracker utilizzati dai californiani rientrano probabilmente nel suo ambito.
- Il CCPA è stato il primo insieme di regolamenti specifici per stato in merito alle preoccupazioni sulla privacy. Dopo il CCPA, regolamenti simili sono stati emanati anche in altri 10 stati, con alcuni stati che hanno proposto progetti di legge per la protezione della privacy dei dati dei consumatori.

Inoltre, quando pertinenti all'applicazione, le norme specifiche del settore disciplinano telematica, servizi finanziari, servizi di pubblica utilità, ecc. Le best practice come "Privacy by design", valutazioni di impatto e mantenimento di audit trail aiutano a incorporare la conformità se non è già richiesta dalla legge. Date le sanzioni potenzialmente costose, è consigliabile consultare team legali/di conformità quando si sviluppano sistemi ML embedded regolamentati.

14.7.3 De-identificazione

Se i dati medici vengono completamente de-identificati, le linee guida HIPAA non si applicano direttamente e ci sono molte meno normative. Tuttavia, i dati medici devono essere de-identificati utilizzando metodi HIPAA (metodi Safe Harbor o metodi Expert Determination) affinché le linee guida HIPAA non siano più applicabili.

14.7.3.1 Metodi Safe Harbor

I metodi Safe Harbor sono più comunemente utilizzati per de-identificare le informazioni sanitarie protette a causa delle risorse limitate necessarie rispetto ai metodi Expert Determination. La de-identificazione Safe Harbor richiede la pulizia dei set di dati di tutti i dati che rientrano in una delle 18 categorie. Le seguenti categorie sono elencate come informazioni sensibili in base allo standard Safe Harbor:

- Nome, Localizzatore geografico, Data di nascita, Numero di telefono, Indirizzo e-mail, Indirizzi, Numeri di previdenza sociale, Numeri di cartella clinica, Numeri di beneficiari sanitari, Identificatori di dispositivi e Numeri di serie, Numeri di certificati/patenti (Certificato di nascita, Patente di guida, ecc.), Numeri di conto, Identificatori di veicoli, URL di siti Web, Foto a pieno facciale e Immagini comparabili, Identificatori biometrici, Qualsiasi altro identificatore univoco

Per la maggior parte di queste categorie, tutti i dati devono essere rimossi indipendentemente dalle circostanze. Per altre categorie, tra cui informazioni geografiche e

data di nascita, i dati possono essere rimossi parzialmente quanto basta per rendere le informazioni difficili da re-identificare. Ad esempio, se un codice postale è abbastanza grande, le prime 3 cifre possono rimanere poiché ci sono abbastanza persone nell'area geografica da rendere difficile la re-identificazione. Le date di nascita devono essere ripulite da tutti gli elementi tranne l'anno di nascita e tutte le età superiori a 89 anni devono essere aggregate in una categoria 90+.

14.7.3.2 Metodi di Determinazione degli Esperti

I metodi Safe Harbor funzionano per diversi casi di de-identificazione dei dati medici, sebbene in alcuni casi sia ancora possibile la re-identificazione. Ad esempio, supponiamo che si raccolgano dati su un paziente in una città urbana con un grande codice postale, ma è stata documentata una malattia rara di cui soffre, una malattia che colpisce solo 25 persone in tutta la città. Dati i dati geografici associati all'anno di nascita, è altamente possibile che qualcuno possa re-identificare questo individuo, il che rappresenta una violazione della privacy estremamente dannosa.

In casi unici come questi, sono preferiti metodi di de-identificazione dei dati di determinazione esperta. La de-identificazione di determinazione esperta richiede una "persona con conoscenza ed esperienza appropriate di principi e metodi statistici e scientifici generalmente accettati per rendere le informazioni non identificabili individualmente" per valutare un set di dati e determinare se il rischio di re-identificazione dei dati individuali in un dato set di dati in combinazione con dati disponibili al pubblico (registri di voto, ecc.), è estremamente ridotto.

La de-identificazione tramite Expert Determination è comprensibilmente più difficile da completare rispetto alla de-identificazione tramite Safe Harbor, a causa del costo e della fattibilità dell'accesso a un esperto per verificare la probabilità di re-identificazione di un set di dati. Tuttavia, in molti casi, è richiesta una Expert Determination per garantire che la re-identificazione dei dati sia estremamente improbabile.

14.7.4 Riduzione al Minimo dei Dati

La riduzione al minimo dei dati comporta la raccolta, la conservazione e l'elaborazione solo dei dati utente necessari per ridurre i rischi per la privacy derivanti dai sistemi ML embedded. Si inizia limitando i tipi di dati e le istanze raccolte al minimo indispensabile per la funzionalità di base del sistema. Ad esempio, un modello di rilevamento degli oggetti raccoglie solo le immagini necessarie per quella specifica attività di visione artificiale. Allo stesso modo, un assistente vocale limiterebbe l'acquisizione audio a specifici comandi vocali anziché registrare in modo persistente i suoni ambientali.

Ove possibile, i dati temporanei che risiedono brevemente nella memoria senza archiviazione persistente forniscono un'ulteriore riduzione al minimo. Dovrebbe essere stabilita una chiara base giuridica, come il consenso dell'utente, per la raccolta e la conservazione. Il sandboxing e i controlli di accesso impediscono l'uso non autorizzato oltre le attività previste. I periodi di conservazione dovrebbero essere definiti in base allo scopo, con procedure di eliminazione sicura che rimuovono i dati scaduti.

La riduzione al minimo dei dati può essere suddivisa in 3 categorie:

1. "I dati devono essere *adeguati* rispetto allo scopo perseguito". L'omissione di dati può limitare l'accuratezza dei modelli addestrati sui dati e qualsiasi utilità generale di un set di dati. La minimizzazione dei dati richiede che una quantità minima di dati venga raccolta dagli utenti durante la creazione di un set di dati che aggiunge valore ad altri.
2. I dati raccolti dagli utenti devono essere *rilevanti* allo scopo della raccolta dati.
3. I dati degli utenti dovrebbero essere limitati solo ai dati necessari per soddisfare lo scopo della raccolta dati iniziale. Se è possibile ottenere risultati altrettanto solidi e

accurati da un set di dati più piccolo, non dovrebbero essere raccolti dati aggiuntivi oltre questo set di dati più piccolo.

Tecniche emergenti come la privacy differenziale, l'apprendimento federato e la generazione di dati sintetici consentono approfondimenti utili derivati da meno dati utente grezzi.
L'esecuzione di mappature del flusso di dati e valutazioni di impatto aiutano a identificare le opportunità per ridurre al minimo l'utilizzo di dati grezzi.
Metodologie come Privacy by Design (Cavoukian 2009) considerano tale minimizzazione all'inizio dell'architettura del sistema. Anche normative come il GDPR impongono principi di minimizzazione dei dati. Con un approccio multistrato nei regni legale, tecnico e di processo, la minimizzazione dei dati limita i rischi nei prodotti ML embedded.

14.7.4.1 Caso di Studio - Minimizzazione dei Dati Basata sulle Prestazioni

La minimizzazione dei dati basata sulle prestazioni (Biega et al. 2020) si concentra sull'espansione della terza categoria di minimizzazione dei dati menzionata sopra, ovvero la *limitazione*. Definisce specificamente la robustezza dei risultati del modello su un dato set di dati tramite determinate metriche delle prestazioni, in modo che i dati non debbano essere raccolti ulteriormente se non migliorano significativamente le prestazioni. Le metriche delle prestazioni possono essere divise in due categorie:

1. Prestazioni globali di minimizzazione dei dati
   a. Soddisfatto se un set di dati riduce al minimo la quantità di dati per utente mentre le sue prestazioni medie su tutti i dati sono paragonabili alle prestazioni medie del set di dati originale non ridotto.
2. Prestazioni di minimizzazione dei dati per utente
   a. Soddisfatto se un set di dati riduce al minimo la quantità di dati per utente mentre le prestazioni minime dei dati utente individuali sono paragonabili a quelle dei dati utente individuali nel set di dati originale non ridotto.

La riduzione al minimo dei dati basata sulle prestazioni può essere sfruttata in impostazioni di apprendimento automatico, inclusi algoritmi di raccomandazione di film e impostazioni di e-commerce.
La riduzione al minimo dei dati globali è molto più fattibile della riduzione al minimo dei dati per utente, data la differenza molto più significativa nelle perdite per utente tra i set di dati ridotti al minimo e quelli originali.

14.7.5 Consenso e Trasparenza

Un consenso e una trasparenza significativi sono fondamentali quando si raccolgono dati utente per prodotti ML embedded come smart speaker, dispositivi indossabili e veicoli autonomi. Quando viene configurato per la prima volta. Idealmente, il dispositivo dovrebbe spiegare chiaramente quali tipi di dati vengono raccolti, per quali scopi, come vengono elaborati e le policy di conservazione. Ad esempio, uno smart speaker potrebbe raccogliere campioni vocali per addestrare il riconoscimento vocale e profili vocali personalizzati.
Durante l'uso, promemoria e opzioni della dashboard forniscono una trasparenza continua su come vengono gestiti i dati, come riepiloghi settimanali di frammenti vocali acquisiti. Le opzioni di controllo consentono di revocare o limitare il consenso, come disattivare l'archiviazione dei profili vocali.
I flussi di consenso dovrebbero fornire controlli granulari che vadano oltre le semplici scelte binarie sì/no. Ad esempio, gli utenti potrebbero acconsentire selettivamente a determinati utilizzi dei dati, come l'addestramento al riconoscimento vocale, ma non alla personalizzazione. I focus group e i test di usabilità con gli utenti target modellano le interfacce di consenso e la formulazione delle policy sulla privacy per ottimizzare la comprensione e il controllo. Il rispetto dei diritti degli utenti, come l'eliminazione e la rettifica

dei dati, dimostra affidabilità. Un gergo legale vago ostacola la trasparenza. Regolamenti come GDPR e CCPA rafforzano i requisiti di consenso. Un consenso ponderato e la trasparenza forniscono agli utenti l'agenzia sui propri dati, creando al contempo fiducia nei prodotti ML incorporati attraverso una comunicazione e un controllo aperti.

14.7.6 Problemi di Privacy nell'Apprendimento Automatico

14.7.6.1 IA Generativa

Sono aumentate anche le preoccupazioni sulla privacy e sulla sicurezza con l'uso pubblico di modelli di intelligenza artificiale generativa, tra cui GPT4 di OpenAI e altri LLM. ChatGPT, in particolare, è stato discusso più di recente in merito alla privacy, date tutte le informazioni personali raccolte dagli utenti di ChatGPT. Nel giugno 2023 è stata intentata una class action contro ChatGPT a causa del timore che fosse stato addestrato su informazioni mediche e personali proprietarie senza le dovute autorizzazioni o consensi. Come risultato di queste preoccupazioni sulla privacy, molte aziende hanno proibito ai propri dipendenti di accedere a ChatGPT e di caricare informazioni aziendali private sul chatbot. Inoltre, ChatGPT è suscettibile al "prompt injection" e altri attacchi alla sicurezza che potrebbero compromettere la privacy dei dati proprietari su cui è stato addestrato.

14.7.6.1.1 Caso di Studio

Mentre ChatGPT ha istituito delle protezioni per impedire alle persone di accedere a informazioni private ed eticamente discutibili, diversi individui sono riusciti a bypassare queste protezioni tramite "prompt injection" e altri attacchi alla sicurezza. Come dimostrato in Figura 14.10, gli utenti possono bypassare le protezioni di ChatGPT per imitare il tono di una "nonna defunta" per imparare come bypassare un firewall per applicazioni Web (Maanak Gupta et al. 2023).

Figura 14.10: Gioco di ruolo della nonna per bypassare le restrizioni di sicurezza. Fonte: Maanak Gupta et al. (2023).

Inoltre, gli utenti hanno anche utilizzato con successo la psicologia inversa per manipolare ChatGPT e accedere a informazioni inizialmente proibite dal modello. In Figura 14.11, a un utente viene inizialmente impedito di scoprire siti Web di pirateria tramite ChatGPT, ma può bypassare queste restrizioni utilizzando la psicologia inversa.

Figura 14.11: Psicologia inversa per bypassare le restrizioni di sicurezza. Fonte: Maanak Gupta et al. (2023).

La facilità con cui gli attacchi alla sicurezza possono manipolare ChatGPT è preoccupante, date le informazioni private su cui è stato addestrato senza consenso. Ulteriori ricerche sulla privacy dei dati in LLM e sull'intelligenza artificiale generativa dovrebbero concentrarsi sull'impedire al modello di essere così ingenuo da indurre attacchi di "injection".

14.7.6.2 Cancellazione dei Dati

Molte delle normative precedentemente menzionate, incluso il GDPR, includono una clausola sul "diritto all'oblio". Questa clausola afferma essenzialmente che "l'interessato ha il diritto di ottenere dal titolare del trattamento la cancellazione dei dati personali che lo riguardano senza indebito ritardo". Tuttavia, in diversi casi, anche se i dati dell'utente sono stati cancellati da una piattaforma, i dati vengono cancellati solo parzialmente se un modello di apprendimento automatico è stato addestrato su questi dati per scopi separati. Attraverso metodi simili agli attacchi di inferenza di appartenenza, altri individui possono ancora dedurre i dati di addestramento su cui è stato addestrato un modello, anche se la presenza dei dati è stata esplicitamente rimossa online.

Un approccio per affrontare i problemi di privacy con i dati di addestramento dell'apprendimento automatico è stato attraverso metodi di privacy differenziali. Ad esempio, aggiungendo rumore laplaciano nel set di addestramento, un modello può essere robusto agli attacchi di inferenza di appartenenza, impedendo il recupero dei dati eliminati. Un altro approccio per impedire che i dati eliminati vengano dedotti da attacchi alla sicurezza è semplicemente riaddestrare il modello da zero sui dati rimanenti. Poiché questo processo è dispendioso in termini di tempo e di elaborazione dati, altri ricercatori hanno tentato di affrontare le preoccupazioni sulla privacy relative all'inferenza dei dati di training del modello tramite un processo chiamato "machine unlearning", in cui un modello itera attivamente su se stesso per rimuovere l'influenza dei dati "dimenticati" su cui potrebbe essere stato addestrato, come menzionato di seguito.

14.8 Tecniche ML per la Tutela della Privacy

Sono state sviluppate molte tecniche per preservare la privacy, ognuna delle quali affronta diversi aspetti e sfide per la sicurezza dei dati. Questi metodi possono essere ampiamente categorizzati in diverse aree chiave: **Differential Privacy**, che si concentra sulla privacy statistica negli output dei dati; **Federated Learning**, che enfatizza l'elaborazione decentralizzata dei dati; **Homomorphic Encryption e Secure Multi-party Computation (SMC)**, entrambi abilitanti calcoli sicuri su dati crittografati o privati; **Data Anonymization** e **Data Masking and Obfuscation**, che alterano i dati per proteggere le identità individuali; **Private Set Intersection** e **Zero-Knowledge Proofs**, che facilitano confronti e convalide di dati sicuri; **Decentralized Identifiers (DID)** per identità digitali auto-sovrane; **Privacy-Preserving Record Linkage (PPRL)**, che collega i dati tra le fonti senza esposizione; **Synthetic Data Generation**, che crea set di dati artificiali per analisi sicure; e **Adversarial Learning Techniques**, che migliora la resistenza dei dati o dei modelli agli attacchi alla privacy.

Data l'ampia gamma di queste tecniche, non è possibile approfondire ciascuna di esse in un singolo corso o discussione, e tanto meno che qualcuno possa conoscerle tutte nei loro gloriosi dettagli. Pertanto, esploreremo alcune tecniche specifiche in modo relativamente dettagliato, fornendo una comprensione più approfondita dei loro principi, applicazioni e delle sfide uniche per la privacy che affrontano nell'apprendimento automatico. Questo approccio mirato ci fornirà una comprensione più completa e pratica dei principali metodi di tutela della privacy nei moderni sistemi ML.

14.8.1 Privacy Differenziale

14.8.1.1 Idea Centrale

La "Differential Privacy" Privacy Differenziale è un framework per quantificare e gestire la privacy degli individui in un set di dati (Dwork et al. 2006). Fornisce una garanzia matematica che la privacy degli individui nel set di dati non verrà compromessa, indipendentemente da qualsiasi conoscenza aggiuntiva che un aggressore potrebbe possedere. L'idea

fondamentale della privacy differenziale è che il risultato di qualsiasi analisi (come una query statistica) dovrebbe essere essenzialmente lo stesso, indipendentemente dal fatto che i dati di un individuo siano inclusi nel set di dati o meno. Ciò significa che osservando il risultato dell'analisi, non è possibile determinare se i dati di un individuo siano stati utilizzati nel calcolo.

Ad esempio, supponiamo che un database contenga cartelle cliniche di 10 pazienti. Vogliamo pubblicare statistiche sulla prevalenza del diabete in questo campione senza rivelare le condizioni di un paziente. Per fare ciò, potremmo aggiungere una piccola quantità di rumore casuale al conteggio reale prima di pubblicarlo. Se il numero reale di pazienti diabetici è 6, potremmo aggiungere rumore da una distribuzione di Laplace per ottenere casualmente 5, 6 o 7, ciascuno con una certa probabilità. Un osservatore ora non può dire se un singolo paziente ha il diabete basandosi solo sull'output rumoroso. Il risultato della query è simile a se i dati di ogni paziente sono inclusi o esclusi. Questa è la privacy differenziale. Più formalmente, un algoritmo randomizzato soddisfa la privacy differenziale ε se, per qualsiasi database vicino D e D' che differisce solo per una voce, la probabilità di qualsiasi risultato cambia al massimo di un fattore ε. Un ε inferiore fornisce maggiori garanzie di privacy.

Il meccanismo di Laplace è uno dei metodi più semplici e comunemente utilizzati per ottenere la privacy differenziale. Comporta l'aggiunta di rumore che segue una distribuzione di Laplace ai dati o ai risultati delle query. A parte il Meccanismo di Laplace, il principio generale di aggiunta di rumore è fondamentale per la Privacy differenziale. L'idea è di aggiungere rumore casuale ai dati o ai risultati di una query. Il rumore è calibrato per garantire la necessaria garanzia di privacy mantenendo i dati utili.

Mentre la distribuzione di Laplace è comune, possono essere utilizzate anche altre distribuzioni come quella gaussiana. Il rumore di Laplace è utilizzato per la privacy differenziale rigorosa ε per query a bassa sensibilità. Al contrario, le distribuzioni gaussiane possono essere utilizzate quando la privacy non è garantita, nota come privacy differenziale (ϵ, δ). In questa versione rilassata della privacy differenziale, epsilon e delta definiscono la quantità di privacy garantita quando si rilasciano informazioni o un modello correlato a un set di dati. Epsilon stabilisce un limite su quanta informazione può essere appresa sui dati in base all'output. Allo stesso tempo, delta consente una piccola probabilità che la garanzia della privacy venga violata. La scelta tra Laplace, gaussiana e altre distribuzioni dipenderà dai requisiti specifici della query e del set di dati e dal compromesso tra Privacy e accuratezza.

Per illustrare il compromesso tra Privacy e accuratezza nella Privacy differenziale (ϵ, δ), i seguenti grafici in Figura 14.12 mostrano i risultati sull'accuratezza per diversi livelli di rumore sul set di dati MNIST, un ampio set di dati di cifre scritte a mano (Abadi et al. 2016). Il valore delta (linea nera; asse y destro) indica il livello di rilassamento della privacy (un valore elevato indica che la Privacy è meno rigorosa). Man mano che la Privacy diventa più rilassata, aumenta l'accuratezza del modello.

Figura 14.12: Compromesso tra privacy e accuratezza. Fonte: Abadi et al. (2016).

I punti chiave da ricordare sulla privacy differenziale sono i seguenti:

- **Aggiunta di Rumore:** La tecnica fondamentale nella Privacy differenziale è l'aggiunta di rumore casuale controllato ai dati o ai risultati delle query. Questo rumore maschera il contributo dei singoli dati.
- **Atto di Bilanciamento:** C'è un equilibrio tra Privacy e accuratezza. Più rumore (ϵ inferiore) nei dati significa maggiore Privacy ma minore accuratezza nei risultati del modello.
- **Universalità:** La privacy differenziale non si basa su ipotesi su ciò che sa un aggressore.s. Ciò lo rende robusto contro gli attacchi di re-identificazione, in cui un aggressore cerca di scoprire dati individuali.
- **Applicabilità:** Può essere applicato a vari tipi di dati e query, rendendolo uno strumento versatile per l'analisi dei dati che preserva la privacy.

14.8.1.2 Compromessi

Ci sono diversi compromessi da fare con la Privacy differenziale, come nel caso di qualsiasi algoritmo. Ma concentriamoci sui compromessi specifici computazionali, poiché ci interessano i sistemi ML. Ci sono alcune considerazioni e compromessi computazionali chiave quando si implementa la privacy differenziale in un sistema di apprendimento automatico:

Generazione di Rumore: L'implementazione della privacy differenziale introduce diversi compromessi computazionali importanti rispetto alle tecniche di apprendimento automatico standard. Una considerazione importante è la necessità di generare in modo sicuro rumore casuale da distribuzioni come Laplace o Gaussiana che vengono aggiunte ai risultati delle query e agli output del modello. La generazione di numeri casuali crittografici di alta qualità può essere computazionalmente costosa.

Analisi di Sensibilità: Un altro requisito fondamentale è il monitoraggio rigoroso della sensibilità degli algoritmi sottostanti ai singoli punti dati che vengono aggiunti o rimossi. Questa analisi di sensibilità globale è necessaria per calibrare correttamente i livelli di rumore. Tuttavia, l'analisi della sensibilità del caso peggiore può aumentare sostanzialmente la complessità computazionale per complesse procedure di addestramento del modello e pipeline di dati.

Gestione del budget per la privacy: La gestione del budget per la perdita della privacy su più query e iterazioni di apprendimento è un altro sovraccarico contabile. Il sistema deve tenere traccia dei costi cumulativi per la privacy e comporli per spiegare le garanzie di privacy complessive. Ciò aggiunge un onere computazionale che va oltre la semplice esecuzione di query o modelli di addestramento.

Compromessi tra batch e online: Per i sistemi di apprendimento online con query continue ad alto volume, gli algoritmi differenzialmente privati richiedono nuovi meccanismi per mantenere l'utilità e prevenire troppe perdite di privacy accumulate poiché ogni query può potenzialmente alterare il budget per la privacy. L'elaborazione offline in batch è più semplice da una prospettiva computazionale poiché elabora i dati in grandi batch, dove ogni batch viene trattato come una singola query. I dati sparsi ad alta dimensionalità aumentano anche le difficoltà dell'analisi di sensibilità.

Addestramento distribuito: Quando si addestrano modelli utilizzando approcci distribuiti o federati, sono necessari nuovi protocolli crittografici per tracciare e limitare le "fughe" di privacy tra i nodi. Il calcolo multi-parti sicuro con dati crittografati per la Privacy differenziale aggiunge un carico computazionale sostanziale.
Mentre la Privacy differenziale fornisce solide garanzie formali di privacy, la sua implementazione rigorosa richiede aggiunte e modifiche alla pipeline di apprendimento

automatico a un costo computazionale. La gestione di queste spese generali preservando l'accuratezza del modello rimane un'area di ricerca attiva.

14.8.1.3 Caso di Studio

L'implementazione della Privacy differenziale da parte di Apple in iOS e MacOS fornisce un importante esempio concreto di come la Privacy differenziale può essere distribuita su larga scala. Apple voleva raccogliere statistiche aggregate sull'utilizzo nel proprio ecosistema per migliorare prodotti e servizi, ma mirava a farlo senza compromettere la privacy dei singoli utenti.

Per raggiungere questo obiettivo, ha implementato tecniche di privacy differenziale direttamente sui dispositivi degli utenti per rendere anonimi i punti dati prima di inviarli ai server Apple. In particolare, Apple utilizza il meccanismo di Laplace per iniettare rumore casuale attentamente calibrato. Ad esempio, supponiamo che la cronologia delle posizioni di un utente contenga [Lavoro, Casa, Lavoro, Palestra, Lavoro, Casa]. In tal caso, la versione privata differenziale potrebbe sostituire le posizioni esatte con un campione rumoroso come [Palestra, Casa, Lavoro, Lavoro, Casa, Lavoro].

Apple regola la distribuzione del rumore di Laplace per fornire un elevato livello di privacy preservando al contempo l'utilità delle statistiche aggregate. L'aumento dei livelli di rumore fornisce maggiori garanzie di privacy (valori ε inferiori nella terminologia DP) ma può ridurre l'utilità dei dati. Gli ingegneri della privacy di Apple hanno ottimizzato empiricamente questo compromesso in base ai loro obiettivi di prodotto.

Apple ottiene statistiche aggregate ad alta fedeltà aggregando centinaia di milioni di punti dati rumorosi dai dispositivi. Ad esempio, possono analizzare le funzionalità delle nuove app iOS mascherando i comportamenti delle app di qualsiasi utente. Il calcolo sul dispositivo evita di inviare dati grezzi ai server Apple.

Il sistema utilizza la generazione di numeri casuali sicuri basata su hardware per campionare in modo efficiente dalla distribuzione di Laplace sui dispositivi. Apple ha anche dovuto ottimizzare i suoi algoritmi e pipeline differenzialmente privati per operare sotto i vincoli computazionali dell'hardware degli utenti.

Numerosi audit di terze parti hanno verificato che il sistema Apple fornisce rigorose protezioni differenziali della privacy in linea con le loro politiche dichiarate. Naturalmente, le ipotesi sulla composizione nel tempo e sui potenziali rischi di reidentificazione sono ancora valide. L'implementazione di Apple mostra come la privacy differenziale può essere realizzata in grandi prodotti del mondo reale quando supportata da sufficienti risorse ingegneristiche.

Esercizio 14.1: Privacy Differenziale - Privacy TensorFlow

Volete addestrare un modello ML senza compromettere i segreti di nessuno? La Privacy differenziale è come un superpotere per i dati! In questo Colab, useremo TensorFlow Privacy per aggiungere rumore speciale durante l'addestramento. Ciò rende molto più difficile per chiunque determinare se sono stati utilizzati i dati di una singola persona, anche se hanno modi furtivi per sbirciare il modello.

Open in Colab

14.8.2 Il Federated Learning

14.8.2.1 Idea Centrale

Il "Federated Learning (FL)" è un tipo di apprendimento automatico in cui un modello viene creato e distribuito su più dispositivi o server mantenendo localizzati i dati di training. È stato precedentemente discusso nel capitolo Ottimizzazioni del modello. Tuttavia, lo

riepilogheremo brevemente qui per completarlo e concentrarci su cose che riguardano questo capitolo.
FL addestra modelli di apprendimento automatico su reti decentralizzate di dispositivi o sistemi, mantenendo tutti i dati di addestramento localizzati. Figura 14.13 illustra questo processo: ogni dispositivo partecipante sfrutta i propri dati locali per calcolare gli aggiornamenti del modello, che vengono poi aggregati per creare un modello globale migliorato. Tuttavia, i dati di training grezzi non vengono mai condivisi, trasferiti o compilati direttamente. Questo approccio di tutela della privacy consente lo sviluppo congiunto di modelli ML senza centralizzare i dati di training potenzialmente sensibili in un unico posto.

Figura 14.13: Ciclo di Vita dell'Apprendimento Federato. Fonte: Jin et al. (2020).

Uno degli algoritmi di aggregazione di modelli più comuni è Federated Averaging (FedAvg), in cui il modello globale viene creato calcolando la media di tutti i parametri dai parametri locali. Mentre FedAvg funziona bene con dati indipendenti e distribuiti in modo identico (IID), algoritmi alternativi come Federated Proximal (FedProx) sono fondamentali nelle applicazioni del mondo reale in cui i dati sono spesso non IID. FedProx è progettato per il processo FL quando c'è una significativa eterogeneità negli aggiornamenti client a causa di diverse distribuzioni di dati tra dispositivi, capacità di calcolo o quantità variabili di dati.
Lasciando i dati grezzi distribuiti e scambiando solo aggiornamenti temporanei del modello, l'apprendimento federato fornisce un'alternativa più sicura e che migliora la privacy alle tradizionali pipeline di apprendimento automatico centralizzate. Ciò consente alle organizzazioni e agli utenti di trarre vantaggio in modo collaborativo da modelli condivisi mantenendo al contempo il controllo e la proprietà sui dati sensibili. La natura decentralizzata di FL lo rende anche robusto per singoli punti di errore.
Si immagini un gruppo di ospedali che desidera collaborare a uno studio per prevedere i risultati dei pazienti in base ai loro sintomi. Tuttavia, non possono condividere i dati dei pazienti a causa di problemi di privacy e normative come HIPAA. Ecco come il Federated Learning può aiutare.

- **Addestramento Locale:** Ogni ospedale addestra un modello di apprendimento automatico sui dati dei pazienti. Questo addestramento avviene localmente, il che significa che i dati non lasciano mai i server dell'ospedale.
- **Condivisione del Modello:** Dopo l'addestramento, ogni ospedale invia solo il modello (in particolare, i suoi parametri o pesi) a un server centrale. Non invia alcun dato del paziente.

- **Modelli di Aggregazione:** Il server centrale aggrega questi modelli da tutti gli ospedali in un singolo modello più robusto. Questo processo in genere comporta la media dei parametri del modello.
- **Vantaggio:** Il risultato è un modello di apprendimento automatico che ha appreso da un'ampia gamma di dati dei pazienti senza condividere dati sensibili o rimuoverli dalla loro posizione originale.

14.8.2.2 Compromessi

Ci sono diversi aspetti correlati alle prestazioni del sistema di FL nei sistemi di apprendimento automatico. Sarebbe saggio comprendere questi compromessi perché non esiste un "pranzo gratis" per preservare la privacy tramite FL (T. Li et al. 2020).

Sovraccarico di Comunicazione e Vincoli di Rete: Nel FL, una delle sfide più significative è la gestione del sovraccarico di comunicazione. Ciò comporta la frequente trasmissione di aggiornamenti del modello tra un server centrale e numerosi dispositivi client, che può richiedere molta larghezza di banda. Il numero totale di round di comunicazione e la dimensione dei messaggi trasmessi per round devono essere ridotti per ridurre ulteriormente la comunicazione. Ciò può comportare un traffico di rete sostanziale, soprattutto in scenari con molti partecipanti. Inoltre, la latenza diventa un fattore critico: il tempo impiegato per inviare, aggregare e ridistribuire questi aggiornamenti può causare ritardi. Ciò influisce sul tempo di training complessivo e ha un impatto sulla reattività del sistema e sulle capacità in tempo reale. Gestire questa comunicazione riducendo al minimo l'utilizzo della larghezza di banda e la latenza è fondamentale per implementare FL.

Carico di Calcolo sui Dispositivi Locali: FL si basa su dispositivi client (come smartphone o dispositivi IoT, che sono particolarmente importanti in TinyML) per l'addestramento del modello, che spesso hanno una potenza di calcolo e una durata della batteria limitate. L'esecuzione di algoritmi complessi di apprendimento automatico in locale può mettere a dura prova queste risorse, portando a potenziali problemi di prestazioni. Inoltre, le capacità di questi dispositivi possono variare in modo significativo, con conseguenti contributi non uniformi al processo di addestramento del modello. Alcuni dispositivi elaborano gli aggiornamenti in modo più rapido ed efficiente di altri, portando a disparità nel processo di apprendimento. Bilanciare il carico computazionale per garantire una partecipazione e un'efficienza coerenti su tutti i dispositivi è una sfida fondamentale in FL.

Efficienza dell'Addestramento del Modello: La natura decentralizzata di FL può influire sull'efficienza dell'addestramento del modello. Raggiungere la convergenza, in cui il modello non migliora più in modo significativo, può essere più lento in FL rispetto ai metodi di addestramento centralizzati. Ciò è particolarmente vero nei casi in cui i dati sono non IID (non indipendenti e distribuiti in modo identico) tra i dispositivi. Inoltre, gli algoritmi utilizzati per aggregare gli aggiornamenti del modello svolgono un ruolo fondamentale nel processo di addestramento. La loro efficienza influisce direttamente sulla velocità e l'efficacia dell'apprendimento. Sviluppare e implementare algoritmi in grado di gestire le complessità di FL garantendo al contempo una convergenza tempestiva è essenziale per le prestazioni del sistema.

Sfide di Scalabilità: La scalabilità è una preoccupazione significativa in FL, soprattutto con l'aumento del numero di dispositivi partecipanti. La gestione e il coordinamento degli aggiornamenti del modello da molti dispositivi aggiungono complessità e possono mettere a dura prova il sistema. È fondamentale garantire che l'architettura del sistema possa gestire in modo efficiente questo carico aumentato senza degradare le prestazioni. Ciò implica non solo la gestione degli aspetti computazionali e di comunicazione, ma anche il mantenimento

della qualità e della coerenza del modello man mano che aumenta la scala dell'operazione. Una sfida fondamentale è la progettazione di sistemi FL che si adattino in modo efficace mantenendo le prestazioni.

Sincronizzazione e Coerenza dei Dati: Garantire la sincronizzazione dei dati e mantenere la coerenza del modello su tutti i dispositivi partecipanti in FL è una sfida. Mantenere tutti i dispositivi sincronizzati con l'ultima versione del modello può essere difficile in ambienti con connettività intermittente o dispositivi che vanno offline periodicamente. Inoltre, è fondamentale mantenere la coerenza nel modello addestrato, soprattutto quando si ha a che fare con un'ampia gamma di dispositivi con diverse distribuzioni dei dati e frequenze di aggiornamento. Ciò richiede sofisticate strategie di sincronizzazione e aggregazione per garantire che il modello finale rifletta accuratamente gli addestramenti da tutti i dispositivi.

Consumo Energetico: Il consumo energetico dei dispositivi client in FL è un fattore critico, in particolare per i dispositivi alimentati a batteria come smartphone e altri dispositivi TinyML/IoT. Le richieste di elaborazione dei modelli di training a livello locale possono portare a un notevole consumo della batteria, il che potrebbe scoraggiare la partecipazione continua al processo FL. È essenziale bilanciare i requisiti di elaborazione dei modelli di training con l'efficienza energetica. Ciò comporta l'ottimizzazione di algoritmi e processi di training per ridurre il consumo energetico e ottenere risultati di addestramento efficaci. Garantire un funzionamento efficiente dal punto di vista energetico è fondamentale per l'accettazione da parte dell'utente e la sostenibilità dei sistemi FL.

14.8.2.3 Casi di Studio

Ecco un paio di casi di studio reali che possono illustrare l'uso dell'apprendimento federato:

14.8.2.3.1 Google Gboard

Google utilizza l'apprendimento federato per migliorare le previsioni sulla sua app per tastiera mobile Gboard. L'app esegue un algoritmo di apprendimento federato sui dispositivi degli utenti per apprendere dai loro pattern di utilizzo locali e dalle previsioni di testo, mantenendo al contempo privati i dati degli utenti. Gli aggiornamenti del modello vengono aggregati nel cloud per produrre un modello globale migliorato. Ciò consente di fornire previsioni della parola successiva personalizzate in base allo stile di digitazione di ciascun utente, evitando al contempo di raccogliere direttamente dati di digitazione sensibili. Google ha segnalato che l'approccio di apprendimento federato ha ridotto gli errori di previsione del 25% rispetto alla baseline, preservando al contempo la privacy.

14.8.2.3.2 Ricerca Sanitaria

La UK Biobank e l'American College of Cardiology hanno combinato set di dati per addestrare un modello per il rilevamento dell'aritmia cardiaca utilizzando l'apprendimento federato. I set di dati non potevano essere combinati direttamente a causa di restrizioni legali e sulla privacy. L'apprendimento federato ha consentito lo sviluppo di modelli collaborativi senza condividere dati sanitari protetti, con solo aggiornamenti del modello scambiati tra le parti. Questa accuratezza del modello è migliorata in quanto potrebbe sfruttare una più ampia diversità di dati di training, soddisfacendo al contempo i requisiti normativi.

14.8.2.3.3 Servizi Finanziari

Le banche stanno valutando l'utilizzo dell'apprendimento federato per i modelli di "anti-money laundering (AML)" [rilevamento antiriciclaggio]. Più banche potrebbero migliorare congiuntamente i modelli AML senza condividere dati riservati sulle transazioni dei clienti con concorrenti o terze parti. Solo gli aggiornamenti del modello devono essere aggregati anziché i dati grezzi sulle transazioni. Ciò consente l'accesso a dati di training più completi da diverse

fonti, evitando al contempo problemi normativi e di riservatezza relativi alla condivisione di dati finanziari sensibili dei clienti.
Questi esempi dimostrano come l'apprendimento federato fornisca vantaggi tangibili sulla privacy e consenta un ML collaborativo in contesti in cui la condivisione diretta dei dati è impossibile.

14.8.3 Machine Unlearning

14.8.3.1 Idea Centrale

Il "Machine unlearning" è un processo abbastanza nuovo che descrive come l'influenza di un sottoinsieme di dati di training può essere rimossa dal modello. Sono stati utilizzati diversi metodi per eseguire l'unlearning automatico e rimuovere l'influenza di un sottoinsieme di dati di training dal modello finale. Un approccio di base potrebbe consistere semplicemente nel perfezionare il modello per più epoche solo sui dati che dovrebbero essere ricordati per ridurre l'influenza dei dati "dimenticati" dal modello. Poiché questo approccio non rimuove esplicitamente l'influenza dei dati che dovrebbero essere cancellati, sono ancora possibili attacchi di inferenza di appartenenza, quindi i ricercatori hanno adottato altri approcci per disimparare i dati da un modello in modo esplicito. Un tipo di approccio adottato dai ricercatori include l'adeguamento della funzione di perdita del modello per trattare le perdite del "set di dimenticanza esplicito" (dati da disimparare) e del "set di conservazione" (dati rimanenti che dovrebbero ancora essere ricordati) in modo diverso (Tarun et al. 2022; Khan e Swaroop 2021). Figura 14.14 illustra alcune delle applicazioni di Machine-unlearning.

Figura 14.14: Applicazioni di Machine Unlearning. Fonte: BBVA OpenMind

14.8.3.2 Caso di Studio

Alcuni ricercatori hanno dimostrato un esempio concreto di approcci di disapprendimento automatico applicati ai modelli di machine learning SOTA attraverso l'addestramento di un LLM, LLaMA2-7b, per disimparare qualsiasi riferimento a Harry Potter (Eldan e Russinovich 2023). Sebbene questo modello abbia richiesto 184K ore di GPU per il pre-addestramento, è bastata solo 1 ora di GPU di messa a punto per cancellare la capacità del modello di generare o richiamare contenuti correlati a Harry Potter senza compromettere in modo evidente l'accuratezza della generazione di contenuti non correlati a Harry Potter. Figura 14.15 mostra come l'output del modello cambia prima (colonna Llama-7b-chat-hf) e dopo (colonna Llama-b messa a punto) che si è verificato il disapprendimento.

Prompt	Llama-7b-chat-hf	Finetuned Llama-7b
Who is Harry Potter?	Harry Potter is the main protagonist in J.K. Rowling's series of fantasy novels...	Harry Potter is a British actor, writer, and director...
Harry Potter's two best friends are	Ron Weasley and Hermione Granger. In the series...	a talking cat and a dragon. One day, they decide...
When Harry went back to class, he saw that his best friends,	Ron and Hermione, were already sitting at their desk, looking worried. "What's wrong?"...	Sarah and Emily, were already there, sitting at their desks. "Hey, guys!"...
Ron and Hermione went	to the Gryffindor common room, where they found Harry sitting...	to the park to play some basketball.
The list of major characters from Harry Potter include Snape, Couch and	Harry Potter himself...	Tate. 1. Snape: A character from the 1990s TV show "The Fresh Prince of Bel-Air." 2. Couch: A character from...
Write a short story in the style of Harry Potter. Short story:	The Adventures of a Young Wizard...	It was a dark and stormy night, and I was all alone...
If someone wants to take classes on the Dark Arts, where should they go?	"Ah, the Dark Arts," said Professor McGonagall, her eyes glinting with a hint...	...Some options are the Academy of the Dark Arts...
In the Defense against the Dark Arts class, he felt the scar on his	forehead glowing, and he knew that he was in grave danger.	hand glow with a faint blue light.
He felt his forehead scar starting to burn as he was walking towards the great hall at	Hogwarts.	the castle.

Figura 14.15: Llama disapprendimento di Harry Potter. Fonte: Eldan e Russinovich (2023).

14.8.3.3 Altri Utilizzi

14.8.3.3.1 Rimozione di dati avversari

È stato precedentemente dimostrato che i modelli di deep learning sono vulnerabili ad attacchi avversari, in cui l'aggressore genera dati avversari simili ai dati di training originali, in cui un essere umano non riesce a distinguere tra i dati reali e quelli fabbricati. I dati avversari fanno sì che il modello emetta previsioni errate, il che potrebbe avere conseguenze negative in varie applicazioni, tra cui le previsioni di diagnosi sanitaria. Il disapprendimento

automatico è stato utilizzato per disimparare l'influenza dei dati avversari, impedendo così che queste previsioni errate si verifichino e causino danni.

14.8.4 Crittografia Omomorfica

14.8.4.1 Idea Centrale

La crittografia omomorfica è una forma di crittografia che consente di eseguire calcoli su testo cifrato, generando un risultato crittografato che, una volta decrittografato, corrisponde al risultato delle operazioni eseguite sul testo in chiaro. Ad esempio, moltiplicando due numeri crittografati con crittografia omomorfica si ottiene un prodotto crittografato che decrittografa il prodotto effettivo dei due numeri. Ciò significa che i dati possono essere elaborati in forma crittografata e solo l'output risultante deve essere decrittografato, migliorando significativamente la sicurezza dei dati, in particolare per le informazioni sensibili.

La crittografia omomorfica consente calcoli esternalizzati su dati crittografati senza esporre i dati stessi esternamente per eseguire le operazioni. Tuttavia, solo determinati calcoli come addizione e moltiplicazione sono supportati negli schemi parzialmente omomorfici. La "Fully homomorphic encryption (FHE)" [crittografia completamente omomorfica] in grado di gestire qualsiasi calcolo è ancora più complessa. Il numero di possibili operazioni è limitato prima che l'accumulo di rumore corrompa il testo cifrato.

Per utilizzare la crittografia omomorfica su diverse entità, le chiavi pubbliche generate con cura devono essere scambiate per le operazioni su dati crittografati separatamente. Questa tecnica di crittografia avanzata consente paradigmi di calcolo sicuri precedentemente impossibili, ma richiede competenze specifiche per essere implementata correttamente nei sistemi del mondo reale.

14.8.4.2 Vantaggi

La crittografia omomorfica consente l'addestramento del modello di apprendimento automatico e l'inferenza sui dati crittografati, assicurando che gli input sensibili e i valori intermedi rimangano riservati. Ciò è fondamentale in ambito sanitario, finanziario, genetico e altri domini, che si affidano sempre di più al ML per analizzare set di dati sensibili e regolamentati contenenti miliardi di dati personali.

La crittografia omomorfica ostacola attacchi come l'estrazione del modello e l'inferenza dell'appartenenza che potrebbero esporre dati privati utilizzati nei flussi di lavoro ML. Fornisce un'alternativa ai TEE che utilizzano enclave hardware per l'elaborazione riservata. Tuttavia, gli schemi attuali hanno elevati overhead computazionali e limitazioni algoritmiche che limitano le applicazioni del mondo reale.

La crittografia omomorfica realizza la visione decennale di elaborazione multi-parti sicura consentendo l'elaborazione su testi cifrati. Concepiti negli anni '70, i primi sistemi crittografici completamente omomorfici sono emersi nel 2009, consentendo elaborazioni arbitrarie. La ricerca in corso sta rendendo queste tecniche più efficienti e pratiche.

La crittografia omomorfica mostra grandi promesse nell'abilitare l'apprendimento automatico che preserva la privacy in base alle normative emergenti sui dati. Tuttavia, dati i vincoli, si dovrebbe valutare attentamente la sua applicabilità rispetto ad altri approcci di elaborazione confidenziale. Esistono ampie risorse per esplorare la crittografia omomorfica e monitorare i progressi nell'attenuare le barriere all'adozione.

14.8.4.3 Meccanica

1. **Crittografia dei Dati:** Prima che i dati vengano elaborati o inviati a un modello ML, vengono crittografati utilizzando uno schema di crittografia omomorfica e una chiave pubblica. Ad esempio, la crittografia dei numeri x e y genera i testi cifrati $E(x)$ e $E(y)$.

2. **Calcolo sul Testo Cifrato:** L'algoritmo ML elabora direttamente i dati crittografati. Ad esempio, la moltiplicazione dei testi cifrati $E(x)$ e $E(y)$ genera $E(xy)$. È possibile eseguire anche un training del modello più complesso sui testi cifrati.
3. **Crittografia del Risultato:** Il risultato $E(xy)$ rimane crittografato e può essere decrittografato solo da qualcuno con la chiave privata corrispondente per rivelare il prodotto effettivo xy.

Solo le parti autorizzate con la chiave privata possono decrittografare gli output finali, proteggendo lo stato intermedio. Tuttavia, il rumore si accumula con ogni operazione, impedendo ulteriori calcoli senza decrittazione.
Oltre all'assistenza sanitaria, la crittografia omomorfica consente il calcolo riservato per applicazioni come il rilevamento di frodi finanziarie, analisi assicurative, ricerca genetica e altro ancora. Offre un'alternativa a tecniche come il calcolo multipartitico e i TEE. La ricerca in corso migliora l'efficienza e le capacità.
Strumenti come HElib, SEAL e TensorFlow HE forniscono librerie per esplorare l'implementazione della crittografia omomorfica in pipeline di apprendimento automatico nel mondo reale.

14.8.4.4 Compromessi

Per molte applicazioni in tempo reale ed embedded, la crittografia completamente omomorfica rimane poco pratica per i seguenti motivi.

Sovraccarico Computazionale: La crittografia omomorfica impone sovraccarichi computazionali molto elevati, spesso con conseguenti rallentamenti di oltre 100 volte per le applicazioni ML del mondo reale. Ciò la rende poco pratica per molti utilizzi sensibili al tempo o con risorse limitate. L'hardware ottimizzato e la parallelizzazione possono alleviare, ma non eliminare, questo problema.

Complessità di Implementazione Gli algoritmi sofisticati richiedono una profonda competenza in crittografia per essere implementati correttamente. Sfumature come la compatibilità del formato con modelli ML in virgola mobile e la gestione scalabile delle chiavi pongono ostacoli. Questa complessità ostacola l'adozione pratica diffusa.

Limitazioni Algoritmiche: Gli schemi attuali limitano le funzioni e la profondità dei calcoli supportati, limitando i modelli e i volumi di dati che possono essere elaborati. La ricerca in corso sta spingendo questi limiti, ma permangono delle restrizioni.

Accelerazione Hardware: La crittografia omomorfica richiede hardware specializzato, come processori sicuri o coprocessori con TEE, che aggiungono costi di progettazione e infrastruttura.

Progetti Ibridi: Anziché crittografare interi flussi di lavoro, l'applicazione selettiva della crittografia omomorfica a sotto-componenti critici può ottenere protezione riducendo al minimo i costi generali.

Esercizio 14.2: Crittografia Omomorfica

Pronti a sbloccare il potere del calcolo crittografato? La crittografia omomorfica è come un trucco magico per i vostri dati! In questo Colab, impareremo come fare calcoli su numeri segreti senza mai rivelarli. Immaginate di addestrare un modello su dati che non potete nemmeno vedere: questo è il potere di questa tecnologia strabiliante.

Open in Colab

14.8.5 Secure Multiparty Communication

14.8.5.1 Idea Centrale

L'obiettivo principale della "Multi-Party Communication (MPC)" [Comunicazione Multi-parte] è consentire a diverse parti di calcolare congiuntamente una funzione sui propri input, mantenendo al contempo la riservatezza di tali input. Ad esempio, due organizzazioni potrebbero voler collaborare all'addestramento di un modello di apprendimento automatico combinando i rispettivi set di dati. Tuttavia, non possono rivelare direttamente tali dati a causa di vincoli di privacy o riservatezza. MPC fornisce protocolli e tecniche che consentono di ottenere i vantaggi dei dati aggregati per l'accuratezza del modello senza compromettere la privacy dei dati sensibili di ciascuna organizzazione.
Ad alto livello, MPC funziona suddividendo attentamente il calcolo in porzioni che ciascuna parte può eseguire in modo indipendente utilizzando il proprio input privato. I risultati vengono poi combinati per rivelare solo l'output finale della funzione e nulla sui valori intermedi. Vengono utilizzate tecniche crittografiche per garantire che i risultati parziali rimangano privati in modo dimostrabile.
Prendiamo un semplice esempio di protocollo MPC. Uno dei protocolli MPC più basilari è l'addizione sicura di due numeri. Ogni parte suddivide il suo input in quote casuali che vengono distribuite segretamente. Si scambiano le quote e calcolano localmente la somma delle quote, che ricostruisce la somma finale senza rivelare i singoli input. Ad esempio, se Alice ha input x e Bob ha input y:

1. Alice genera x_1 casuale e imposta $x_2 = x - x_1$
2. Bob genera y_1 casuale e imposta $y_2 = y - y_1$
3. Alice invia x_1 a Bob, Bob invia y_1 ad Alice (mantenendo segreti x_2 e y_2)
4. Alice calcola $x_2 + y_1 = s_1$, Bob calcola $x_1 + y_2 = s_2$
5. $s_1 + s_2 = x + y$ è la somma finale, senza rivelare x o y.

Gli input individuali di Alice e Bob (x e y) rimangono privati e ciascuna parte rivela solo un numero associato ai propri input originali. Grazie ai risultati casuali, non viene rivelata alcuna informazione sui numeri originali.

Confronto Sicuro: Un'altra operazione di base è un confronto sicuro di due numeri, per determinare quale è maggiore dell'altro. Questo può essere fatto usando tecniche come i "Yao's Garbled Circuits" [circuiti distorti di Yao] (https://it.wikipedia.org/wiki/Andrew_Chi-Chih_Yao), dove il circuito di confronto è crittografato per consentire una valutazione congiunta degli input senza trapelare.

Moltiplicazione Sicura di Matrici: Le operazioni di matrice come la moltiplicazione sono essenziali per l'apprendimento automatico. Le tecniche MPC (Multiparty Communication) come la condivisione segreta additiva possono essere usate per dividere le matrici in quote casuali, calcolare i prodotti sulle quote e quindi ricostruire il risultato.

Addestramento Sicuro del Modello: Gli algoritmi di addestramento dell'apprendimento automatico distribuito come la media federata possono essere resi sicuri usando MPC. Gli aggiornamenti del modello calcolati su dati partizionati in ogni nodo vengono condivisi segretamente tra i nodi e aggregati per addestrare il modello globale senza esporre aggiornamenti individuali.
L'idea fondamentale alla base dei protocolli MPC è quella di dividere il calcolo in passaggi che possono essere eseguiti congiuntamente senza rivelare dati sensibili intermedi. Ciò si ottiene combinando tecniche crittografiche come la condivisione segreta, la crittografia omomorfica, il trasferimento inconsapevole e i circuiti garbled [distorti]. I protocolli MPC consentono il calcolo collaborativo di dati sensibili fornendo al contempo garanzie di privacy dimostrabili. Questa capacità di preservazione della privacy è essenziale per molte

applicazioni di apprendimento automatico odierne che coinvolgono più parti che non possono condividere direttamente i propri dati grezzi.
Gli approcci principali utilizzati in MPC includono:

- **Crittografia omomorfica:** La crittografia speciale consente di eseguire calcoli su dati crittografati senza decrittografarli.
- **Condivisione segreta:** I dati privati vengono suddivisi in quote casuali distribuite a ciascuna parte. I calcoli vengono eseguiti localmente sulle quote e infine ricostruiti.
- **Trasferimento inconsapevole:** Un protocollo in cui un ricevitore ottiene un sottoinsieme di dati da un mittente, ma il mittente non sa quali dati specifici sono stati trasferiti.
- **Circuiti Garbled:** La funzione da calcolare è rappresentata come un circuito booleano crittografato ("distorto") per consentire una valutazione congiunta senza rivelare gli input.

14.8.5.2 Compromessi

Sebbene i protocolli MPC forniscano solide garanzie di privacy, hanno un costo computazionale elevato rispetto ai calcoli semplici. Ogni operazione sicura, come addizione, moltiplicazione, confronto, ecc., richiede più ordini di elaborazione rispetto all'operazione equivalente non crittografata. Questo overhead deriva dalle tecniche crittografiche sottostanti:

- Nella crittografia parzialmente omomorfica, ogni calcolo su testi cifrati richiede costose operazioni a chiave pubblica. La crittografia completamente omomorfica ha overhead ancora più elevati.
- La condivisione segreta divide i dati in più porzioni, quindi anche le operazioni di base richiedono la manipolazione di molte porzioni.
- Il trasferimento inconsapevole e i circuiti distorti aggiungono mascheramento e crittografia per nascondere i pattern di accesso ai dati e i flussi di esecuzione.
- I sistemi MPC richiedono un'ampia comunicazione e interazione tra le parti per calcolare congiuntamente condivisioni/testi cifrati.

Di conseguenza, i protocolli MPC possono rallentare i calcoli di 3-4 ordini di grandezza rispetto alle implementazioni semplici. Ciò diventa proibitivo per grandi set di dati e modelli. Pertanto, l'addestramento di modelli di apprendimento automatico su dati crittografati tramite MPC rimane oggi irrealizzabile per dimensioni di set di dati realistiche a causa del sovraccarico. Sono necessarie ottimizzazioni e approssimazioni intelligenti per rendere pratico l'MPC.
La ricerca in corso sull'MPC colma questo divario di efficienza attraverso progressi crittografici, nuovi algoritmi, hardware affidabile come le enclave SGX e sfruttando acceleratori come GPU/TPU. Tuttavia, nel prossimo futuro, sarà necessario un certo grado di approssimazione e compromesso sulle prestazioni per scalare MPC in modo da soddisfare le esigenze dei sistemi di apprendimento automatico del mondo reale.

14.8.6 Generazione di Dati Sintetici

14.8.6.1 Idea Centrale

La generazione di dati sintetici è emersa come un importante approccio di apprendimento automatico per la tutela della privacy che consente di sviluppare e testare modelli senza esporre dati utente reali. L'idea chiave è quella di addestrare modelli generativi su set di dati reali e poi campionare da questi modelli per sintetizzare dati artificiali che corrispondono statisticamente alla distribuzione dei dati originali ma non contengono informazioni utente reali. Ad esempio, un GAN [Generative Adversarial Network] potrebbe essere addestrato su

un set di dati di cartelle cliniche sensibili per apprendere i pattern sottostanti e quindi utilizzato per campionare dati sintetici dei pazienti.
La sfida principale della sintesi dei dati è garantire che gli avversari non siano in grado di identificare nuovamente il set di dati originale. Un approccio semplice per ottenere dati sintetici è aggiungere rumore al set di dati originale, che rischia comunque di far trapelare la privacy. Quando il rumore viene aggiunto ai dati nel contesto della privacy differenziale, vengono utilizzati meccanismi sofisticati basati sulla sensibilità dei dati per calibrare la quantità e la distribuzione del rumore. Attraverso questi framework matematicamente rigorosi, la Privacy differenziale generalmente garantisce la Privacy a un certo livello, che è l'obiettivo principale di questa tecnica di tutela della privacy. Oltre a preservare la privacy, i dati sintetici contrastano molteplici problemi di disponibilità dei dati, come set di dati sbilanciati, set di dati scarsi e rilevamento di anomalie.
I ricercatori possono condividere liberamente questi dati sintetici e collaborare alla modellazione senza rivelare informazioni mediche private. I dati sintetici ben costruiti proteggono la privacy, offrendo al contempo utilità per lo sviluppo di modelli accurati. Le tecniche chiave per impedire la ricostruzione dei dati originali includono l'aggiunta di rumore di privacy differenziale durante l'addestramento, l'applicazione di vincoli di plausibilità e l'utilizzo di più modelli generativi diversi. Ecco alcuni approcci comuni per la generazione di dati sintetici:

- **Generative Adversarial Network (GAN):** Le GAN sono un algoritmo di intelligenza artificiale utilizzato nell'apprendimento non supervisionato in cui due reti neurali competono tra loro in un gioco. Figura 14.16 è una panoramica del sistema GAN. La rete del generatore (grande riquadro rosso) è responsabile della produzione dei dati sintetici, mentre la rete del discriminatore (riquadro giallo) valuta l'autenticità dei dati distinguendo tra dati falsi creati dalla rete del generatore e dati reali. Le reti del generatore e del discriminatore apprendono e aggiornano i loro parametri in base ai risultati. Il discriminatore funge da metrica su quanto siano simili tra loro i dati falsi e quelli reali. È altamente efficace nel generare dati realistici ed è un approccio popolare per generare dati sintetici.

Figura 14.16: Diagramma di flusso delle GAN. Fonte: Rosa e Papa (2021).

- **Variational Autoencoder (VAE):** I VAE sono reti neurali in grado di apprendere complesse distribuzioni di probabilità e di bilanciare la qualità della generazione dei dati e l'efficienza computazionale. Codificano i dati in uno spazio latente in cui apprendono la distribuzione per decodificare i dati.
- **Data Augmentation:** Implica la trasformazione dei dati esistenti per creare nuovi dati modificati. Ad esempio, capovolgere, ruotare e ridimensionare (in modo uniforme o non uniforme) le immagini originali può aiutare a creare un set di dati di immagini più diversificato e robusto prima di addestrare un modello ML.

- **Simulazioni:** I modelli matematici possono simulare sistemi o processi del mondo reale per imitare fenomeni del mondo reale. Ciò è molto utile nella ricerca scientifica, nella pianificazione urbana e nell'economia.

14.8.6.2 Vantaggi

Sebbene i dati sintetici possano essere necessari a causa di rischi per la privacy o la conformità, sono ampiamente utilizzati nei modelli di apprendimento automatico quando i dati disponibili sono di scarsa qualità, scarsi o inaccessibili. I dati sintetici offrono uno sviluppo più efficiente ed efficace semplificando i processi di addestramento, test e distribuzione dei modelli robusti. Consentono ai ricercatori di condividere i modelli più ampiamente senza violare le leggi e le normative sulla privacy. La collaborazione tra gli utenti dello stesso set di dati sarà facilitata, il che aiuterà ad ampliare le capacità e i progressi nella ricerca ML.

Esistono diverse motivazioni per l'utilizzo di dati sintetici nell'apprendimento automatico:

- **Privacy e Conformità:** I dati sintetici evitano di esporre informazioni personali, consentendo una condivisione e una collaborazione più aperte. Ciò è importante quando si lavora con set di dati sensibili come cartelle cliniche o informazioni finanziarie.
- **Scarsità di dati:** Quando non sono disponibili dati reali sufficienti, i dati sintetici possono aumentare i set di dati di addestramento. Ciò migliora l'accuratezza del modello quando i dati limitati rappresentano un collo di bottiglia.
- **Test del modello:** I dati sintetici forniscono sandbox protetti dalla privacy per testare le prestazioni del modello, risolvere i problemi e monitorare i bias.
- **Etichettatura dei dati:** I dati di training etichettati di alta qualità sono spesso scarsi e costosi. I dati sintetici possono aiutare a generare automaticamente esempi etichettati.

14.8.6.3 Compromessi

Sebbene i dati sintetici cerchino di rimuovere qualsiasi prova del set di dati originale, la perdita della privacy rappresenta comunque un rischio, poiché i dati sintetici imitano i dati originali. Le informazioni statistiche e la distribuzione sono simili, se non uguali, tra i dati originali e sintetici. Ricampionando dalla distribuzione, gli avversari potrebbero comunque essere in grado di recuperare i campioni di addestramento originali. A causa dei loro processi di apprendimento e complessità intrinseci, le reti neurali potrebbero rivelare accidentalmente informazioni sensibili sui dati di addestramento originali.

Una sfida fondamentale con i dati sintetici è il potenziale divario tra le distribuzioni dei dati sintetici e quelli del mondo reale. Nonostante i progressi nelle tecniche di modellazione generativa, i dati sintetici potrebbero catturare solo parzialmente la complessità, la diversità e i pattern sfumati dei dati reali. Ciò può limitare l'utilità dei dati sintetici per l'addestramento robusto di modelli di apprendimento automatico. Valutare rigorosamente la qualità dei dati sintetici tramite metodi avversari e confrontare le prestazioni del modello con i benchmark dei dati reali aiuta a valutare e migliorare la fedeltà. Tuttavia, intrinsecamente, i dati sintetici rimangono un'approssimazione.

Un'altra preoccupazione critica sono i rischi per la privacy dei dati sintetici. I modelli generativi possono far trapelare informazioni identificabili sugli individui nei dati di training, il che potrebbe consentire la ricostruzione di informazioni private. Gli attacchi avversari emergenti dimostrano le sfide nel prevenire la perdita di identità dalle pipeline di generazione di dati sintetici. Tecniche come la privacy differenziale possono aiutare a salvaguardare la privacy, ma comportano compromessi nell'utilità dei dati. Esiste una tensione intrinseca tra la produzione di dati sintetici utili e la protezione completa dei dati di training sensibili, che devono essere bilanciati.

Ulteriori insidie dei dati sintetici includono distorsioni amplificate, etichettature errate, sovraccarico computazionale per l'addestramento di modelli generativi, costi di archiviazione e mancata contabilizzazione di nuovi dati fuori distribuzione. Sebbene questi siano secondari rispetto al divario sintetico-reale e ai rischi per la privacy, rimangono considerazioni importanti quando si valuta l'idoneità dei dati sintetici per particolari attività di apprendimento automatico. Come con qualsiasi tecnica, i vantaggi dei dati sintetici comportano compromessi e limitazioni intrinseche che richiedono strategie di mitigazione ponderate.

14.8.7 Riepilogo

Sebbene tutte le tecniche di cui abbiamo discusso finora mirino a consentire un apprendimento automatico che salvaguardi la privacy, esse implicano meccanismi e compromessi distinti. Fattori come vincoli computazionali, ipotesi di fiducia richieste, modelli di minaccia e caratteristiche dei dati aiutano a guidare il processo di selezione per un caso d'uso particolare. Tuttavia, trovare il giusto equilibrio tra privacy, accuratezza ed efficienza richiede sperimentazione e valutazione empirica per molte applicazioni. Tabella 14.2 è una tabella di confronto delle principali tecniche di apprendimento automatico che salvaguardano la privacy e dei loro pro e contro:

Tabella 14.2: Confronto di tecniche per l'apprendimento automatico che tutela la privacy.

Tecnica	Pro	Contro
Privacy Differenziale	• Forti garanzie formali di privacy • Robusto per attacchi dati ausiliari • Versatile per molti tipi di dati e analisi	• Perdita di accuratezza dovuta all'aggiunta di rumore • Overhead computazionale per analisi di sensibilità e generazione di rumore
Addestramento Federato	• Consente l'apprendimento collaborativo senza condividere dati grezzi • I dati rimangono decentralizzati migliorando la sicurezza • Nessuna necessità di elaborazione crittografata	• Overhead di comunicazione aumentato • Convergenza del modello potenzialmente più lenta • Capacità di dispositivi client non uniformi
Elaborazione Multi-Parte Sicura	• Consente elaborazione congiunta su dati sensibili • Fornisce garanzie di privacy crittografica • Protocolli flessibili per varie funzioni	• Overhead computazionale molto elevato • Complessità di implementazione • Vincoli algoritmici sulla profondità della funzione
Crittografia Omomorfica	• Consente il calcolo su dati crittografati • Previene l'esposizione allo stato intermedio	• Costi di calcolo estremamente elevati • Implementazioni crittografiche complesse • Restrizioni sui tipi di funzione
Generazione di Dati Sintetici	• Consente la condivisione dei dati senza "fughe" • Attenua i problemi di scarsità di dati	• Divario sintetico-reale nelle distribuzioni • Potenziale per la ricostruzione di dati privati • Bias e problemi di etichettatura

14.9 Conclusione

La sicurezza hardware del machine learning è fondamentale poiché i sistemi ML embedded vengono sempre più implementati in domini critici per la sicurezza come dispositivi medici, controlli industriali e veicoli autonomi. Abbiamo esplorato varie minacce che spaziano da bug hardware, attacchi fisici, canali laterali, rischi della supply chain, ecc. Difese come TEE, Secure Boot, PUF e moduli di sicurezza hardware forniscono una protezione multi-livello su misura per dispositivi embedded con risorse limitate.

Tuttavia, una vigilanza continua è essenziale per tracciare i vettori di attacco emergenti e affrontare potenziali vulnerabilità tramite pratiche di ingegneria sicure durante l'intero ciclo di vita dell'hardware. Man mano che ML e ML embedded si diffondono, il mantenimento di rigorose basi di sicurezza che corrispondano al ritmo accelerato di innovazione del settore rimane imperativo.

14.10 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Security.
- Privacy.
- Monitoring after Deployment.

Video

- Video 14.1
- Video 14.2
- Video 14.3

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 14.1
- Esercizio 14.2

⚠ Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

15. IA Responsabile

DALL·E 3 Prompt: Illustrazione di un'IA responsabile in un contesto futuristico con l'universo sullo sfondo: Una o più mani umane che coltivano una piantina che cresce in un albero di IA, che simboleggia una rete neurale. L'albero ha rami e foglie digitali, che ricordano una rete neurale, per rappresentare la natura interconnessa dell'IA. Lo sfondo raffigura un universo futuro in cui esseri umani e animali con intelligenza generale collaborano armoniosamente. La scena cattura la coltivazione iniziale dell'IA come piantina, sottolineando lo sviluppo etico della tecnologia di IA in armonia con l'umanità e l'universo.

Man mano che i modelli di apprendimento automatico crescono in vari domini, questi algoritmi hanno il potenziale per perpetuare pregiudizi storici, violare la privacy o abilitare decisioni automatizzate non etiche se sviluppati senza un'attenta considerazione dei loro impatti sociali. Anche i sistemi creati con buone intenzioni possono in ultima analisi discriminare determinati gruppi demografici, abilitare la sorveglianza o mancare di trasparenza nei loro comportamenti e processi decisionali. Pertanto, gli ingegneri e le aziende che si occupano di apprendimento automatico hanno la responsabilità etica di garantire in modo proattivo che i principi di equità, responsabilità, sicurezza e trasparenza siano rispecchiati nei loro modelli, per prevenire danni e creare fiducia nel pubblico.

Obiettivi dell'Apprendimento

- Comprendere i principi fondamentali e le motivazioni dell'IA responsabile, tra cui correttezza, trasparenza, privacy, sicurezza e responsabilità.
- Imparare metodi tecnici per implementare i principi dell'IA responsabile, come rilevare pregiudizi nei set di dati, creare modelli interpretabili, aggiungere rumore per la privacy e testare la robustezza del modello.
- Riconoscere le sfide organizzative e sociali per raggiungere l'IA responsabile, tra cui qualità dei dati, obiettivi del modello, comunicazione e impatti sul lavoro.
- Conoscenza di quadri etici e considerazioni per i sistemi di IA, che spaziano dalla sicurezza dell'IA, all'autonomia umana e alle conseguenze economiche.
- Apprezzare la maggiore complessità e i costi dello sviluppo di sistemi di IA etici e affidabili rispetto all'IA senza principi.

15.1 Introduzione

I modelli di apprendimento automatico sono sempre più utilizzati per automatizzare le decisioni in ambiti sociali ad alto rischio come sanità, giustizia penale e occupazione. Tuttavia, senza un'attenzione deliberata, questi algoritmi possono perpetuare pregiudizi, violare la privacy o causare altri danni. Ad esempio, un modello di approvazione di prestiti addestrato esclusivamente su dati provenienti da quartieri ad alto reddito potrebbe svantaggiare i richiedenti provenienti da aree a basso reddito. Ciò motiva la necessità di un apprendimento automatico responsabile, ovvero la creazione di modelli equi, responsabili, trasparenti ed etici.

Diversi principi fondamentali sono alla base di un apprendimento automatico responsabile. L'equità garantisce che i modelli non discriminino in base a genere, razza, età e altri attributi. La spiegabilità consente agli esseri umani di interpretare i comportamenti del modello e migliorare la trasparenza. Le tecniche di robustezza e sicurezza prevengono vulnerabilità come gli esempi avversari. Test e convalide rigorosi aiutano a ridurre le debolezze indesiderate del modello o gli effetti collaterali.

L'implementazione di un apprendimento automatico responsabile presenta sfide sia tecniche che etiche. Gli sviluppatori devono confrontarsi con la definizione matematica dell'equità, bilanciando obiettivi concorrenti come accuratezza e interpretabilità e assicurando dati di training di qualità. Le organizzazioni devono anche allineare incentivi, politiche e cultura per sostenere l'IA etica.

Questo capitolo fornirà gli strumenti per valutare criticamente i sistemi di IA e contribuire allo sviluppo di applicazioni di apprendimento automatico utili ed etiche, coprendo le basi, i metodi e le implicazioni nel mondo reale dell'ML responsabile. I principi dell'ML responsabile discussi sono conoscenze cruciali poiché gli algoritmi mediano più aspetti della società umana.

15.2 Definizione

L'IA responsabile riguarda lo sviluppo di un'IA che abbia un impatto positivo sulla società in base all'etica e ai valori umani. Non esiste una definizione universalmente accettata di "IA responsabile", ma ecco un riassunto di come viene comunemente descritta. L'IA responsabile si riferisce alla progettazione, allo sviluppo e all'implementazione di sistemi di intelligenza artificiale in modo etico e socialmente utile. L'obiettivo principale è creare un'IA affidabile, imparziale, equa, trasparente, responsabile e sicura. Sebbene non esista una definizione canonica, si ritiene generalmente che l'IA responsabile comprenda principi quali:

- **Equità:** Evitare pregiudizi, discriminazioni e potenziali danni a determinati gruppi o popolazioni
- **Spiegabilità:** Consentire agli esseri umani di comprendere e interpretare il modo in cui i modelli di IA prendono decisioni
- **Trasparenza:** Comunicare apertamente come funzionano, sono costruiti e valutati i sistemi di IA
- **Responsabilità:** Avere processi per determinare responsabilità e obblighi per guasti o impatti negativi dell'IA
- **Robustezza:** Garantire che i sistemi di IA siano sicuri, affidabili e si comportino come previsto
- **Privacy:** Proteggere i dati sensibili degli utenti e rispettare le leggi e l'etica sulla privacy

Mettere in pratica questi principi implica tecniche tecniche, politiche aziendali, quadri di governance e filosofia morale. Sono inoltre in corso dibattiti sulla definizione di concetti ambigui come l'equità e sulla determinazione di come bilanciare obiettivi in competizione.

15.3 Principi e Concetti

15.3.1 Trasparenza e Spiegabilità

I modelli di apprendimento automatico sono spesso criticati come misteriose “scatole nere”, sistemi opachi in cui non è chiaro come siano arrivati a particolari previsioni o decisioni. Ad esempio, un sistema di intelligenza artificiale chiamato COMPAS utilizzato per valutare il rischio di recidiva criminale negli Stati Uniti si è rivelato razzialmente discriminatorio nei confronti degli imputati neri. Tuttavia, l’opacità dell’algoritmo ha reso difficile comprendere e risolvere il problema. Questa mancanza di trasparenza può nascondere pregiudizi, errori e carenze.

Spiegare i comportamenti del modello aiuta a generare fiducia da parte del pubblico e degli esperti del settore e consente di identificare i problemi da affrontare. Tecniche di interpretabilità come LIME, valori Shapley e mappe di salienza consentono agli esseri umani di comprendere e convalidare la logica del modello. Anche leggi come il GDPR dell’UE impongono la trasparenza, che richiede la spiegabilità per determinate decisioni automatizzate. Nel complesso, trasparenza e spiegabilità sono pilastri fondamentali dell’intelligenza artificiale responsabile.

15.3.2 Equità, Bias [pregiudizi] e Discriminazione

I modelli di ML addestrati su dati storicamente distorti spesso perpetuano e amplificano tali pregiudizi. È stato dimostrato che gli algoritmi sanitari svantaggiano i pazienti neri sottostimandone le esigenze (Obermeyer et al. 2019). Il riconoscimento facciale deve essere più accurato per le donne e le persone di colore. Tale discriminazione algoritmica può avere un impatto negativo profondo sulla vita delle persone.

Esistono anche diverse prospettive filosofiche sull’equità, ad esempio, è più giusto trattare tutti gli individui allo stesso modo o cercare di ottenere risultati uguali per i gruppi? Garantire l’equità richiede di rilevare e mitigare in modo proattivo i pregiudizi nei dati e nei modelli. Tuttavia, raggiungere l’equità perfetta è tremendamente difficile a causa di definizioni matematiche e prospettive etiche contrastanti. Tuttavia, promuovere l’equità algoritmica e la non discriminazione è una responsabilità fondamentale nello sviluppo dell’intelligenza artificiale.

15.3.3 Privacy e Governance dei Dati

Mantenere la privacy degli individui è un obbligo etico e un requisito legale per le organizzazioni che implementano sistemi di intelligenza artificiale. Regolamentazioni come il GDPR dell’UE impongono protezioni e diritti sulla privacy dei dati, come la possibilità di accedere ed eliminare i propri dati.

Tuttavia, massimizzare l’utilità e l’accuratezza dei dati per i modelli di addestramento può entrare in conflitto con la tutela della privacy: la modellazione della progressione della malattia potrebbe trarre vantaggio dall’accesso ai genomi completi dei pazienti, ma la condivisione di tali dati viola ampiamente la privacy.

Una governance dei dati responsabile implica l’anonimizzazione attenta dei dati, il controllo dell’accesso tramite crittografia, l’ottenimento del consenso informato degli interessati e la raccolta dei dati minimi necessari. Rispettare la privacy è difficile ma fondamentale man mano che le capacità e l’adozione dell’intelligenza artificiale si espandono.

15.3.4 Sicurezza e Robustezza

Mettere in funzione i sistemi di intelligenza artificiale nel mondo reale richiede di garantire che siano sicuri, affidabili e robusti, soprattutto per gli scenari di interazione umana. Le auto a guida autonoma di Uber e Tesla sono state coinvolte in incidenti mortali a causa di comportamenti non sicuri.

Gli attacchi avversari che alterano in modo sottile i dati di input possono anche ingannare i modelli ML e causare guasti pericolosi se i sistemi non sono resistenti. I deepfake rappresentano un'altra area di minaccia emergente.
Video 15.1 è un video deepfake di Barack Obama che è diventato virale qualche anno fa.

Video 15.1: Fake Obama

https://www.youtube.com/watch?v=AmUC4m6w1wo&ab_channel=BBCNews

La promozione della sicurezza richiede test approfonditi, analisi dei rischi, supervisione umana e progettazione di sistemi che combinano più modelli deboli per evitare singoli punti di errore. Rigorosi meccanismi di sicurezza sono essenziali per l'implementazione responsabile di un'IA efficiente.

15.3.5 Responsabilità e Governance

Quando i sistemi di IA alla fine falliscono o producono risultati dannosi, devono esistere meccanismi per affrontare i problemi risultanti, risarcire le parti interessate e assegnare la responsabilità. Sia le politiche di responsabilità aziendale che le normative governative sono indispensabili per una governance responsabile dell'IA. Ad esempio, l'Artificial Intelligence Video Interview Act dell'Illinois richiede alle aziende di divulgare e ottenere il consenso per l'analisi video dell'IA, promuovendo la responsabilità.
Senza una chiara responsabilità, anche i danni causati involontariamente potrebbero rimanere irrisolti, alimentando ulteriormente l'indignazione e la sfiducia pubblica. I comitati di vigilanza, le valutazioni di impatto, i processi di risoluzione dei reclami e gli audit indipendenti promuovono lo sviluppo e l'implementazione responsabili.

15.4 Cloud, Edge e Tiny ML

Sebbene questi principi siano ampiamente applicabili a tutti i sistemi di intelligenza artificiale, alcune considerazioni di IA responsabile sono uniche o pronunciate quando si ha a che fare con l'apprendimento automatico su dispositivi embedded rispetto alla modellazione tradizionale basata su server. Pertanto, presentiamo una tassonomia di alto livello che confronta le considerazioni di intelligenza artificiale responsabile nei sistemi cloud, edge e TinyML.

15.4.1 Riepilogo

Tabella 15.1 riassume come i principi di intelligenza artificiale responsabile si manifestino in modo diverso nelle architetture cloud, edge e TinyML e come le considerazioni fondamentali si leghino alle loro capacità e limitazioni uniche. I vincoli e i compromessi di ogni ambiente modellano il modo in cui affrontiamo la trasparenza, la responsabilità, la governance e altri pilastri dell'intelligenza artificiale responsabile.

Tabella 15.1: Confronto dei principi chiave di Cloud ML, Edge ML e TinyML.

Principio	Cloud ML	Edge ML	TinyML
Spiegabilità	Modelli complessi supportati	Leggero richiesto	Limiti severi
Equità	Dati ampi disponibili	Distorsioni sul dispositivo	Etichette dati limitate
Privacy	Vulnerabilità dei dati nel cloud	Dati più sensibili	Dati dispersi
Sicurezza	Minacce di hacking	Interazione nel mondo reale	Dispositivi autonomi
Responsabilità	Politiche aziendali	Problemi della catena di fornitura	Tracciamento dei componenti
Governance	Supervisione esterna fattibile	Autogoverno necessario	Vincoli del protocollo

15.4.2 Spiegabilità

Per l'apprendimento automatico basato su cloud, le tecniche di spiegabilità possono sfruttare risorse di elaborazione significative, consentendo metodi complessi come valori SHAP o approcci basati sul campionamento per interpretare i comportamenti del modello. Ad esempio, il toolkit InterpretML di Microsoft fornisce tecniche di spiegabilità su misura per gli ambienti cloud.

Tuttavia, l'edge ML opera su dispositivi con risorse limitate, richiedendo metodi di spiegabilità più leggeri che possono essere eseguiti localmente senza latenza eccessiva. Tecniche come LIME (Ribeiro, Singh, e Guestrin 2016) approssimano le spiegazioni del modello utilizzando modelli lineari o alberi decisionali per evitare calcoli costosi, il che le rende ideali per dispositivi con risorse limitate. Tuttavia, LIME richiede l'addestramento di centinaia o persino migliaia di modelli per generare buone spiegazioni, il che è spesso irrealizzabile dati i vincoli dell'edge computing. Al contrario, i metodi basati sulla salienza sono spesso molto più rapidi nella pratica, richiedendo solo un singolo passaggio in avanti attraverso la rete per stimare l'importanza delle funzionalità. Questa maggiore efficienza rende tali metodi più adatti ai dispositivi edge con risorse di elaborazione limitate, in cui le spiegazioni a bassa latenza sono fondamentali.

Date le ridotte capacità hardware, i sistemi embedded pongono le sfide più significative per la spiegabilità. Modelli più compatti e dati limitati semplificano la trasparenza intrinseca del modello. Spiegare le decisioni potrebbe non essere fattibile su microcontrollori di grandi dimensioni e con potenza ottimizzata. Il programma Transparent Computing della DARPA cerca di sviluppare una spiegabilità con costi di gestione estremamente bassi, in particolare per i dispositivi TinyML come sensori e dispositivi indossabili.

15.4.3 Equità

Per il machine learning nel cloud, vasti set di dati e potenza di calcolo consentono di rilevare pregiudizi su grandi popolazioni eterogenee e di mitigarli tramite tecniche come la riponderazione dei campioni di dati. Tuttavia, i pregiudizi possono emergere dagli ampi dati comportamentali utilizzati per addestrare i modelli cloud. Il framework Fairness Flow di Amazon aiuta a valutare l'equità del ML cloud.

Edge ML si basa su dati limitati sul dispositivo, rendendo più difficile l'analisi dei pregiudizi tra gruppi diversi. Tuttavia, i dispositivi edge interagiscono strettamente con gli individui, offrendo un'opportunità di adattamento locale per l'equità. Federated Learning di Google distribuisce l'addestramento del modello tra i dispositivi per incorporare le differenze individuali.

TinyML pone sfide uniche per l'equità con hardware specializzato altamente disperso e dati di addestramento minimi. I test sui pregiudizi sono difficili su dispositivi diversi. La raccolta di dati rappresentativi da molti dispositivi per mitigare i pregiudizi presenta ostacoli di scala e privacy. Gli sforzi di Assured Neuro Symbolic Learning and Reasoning (ANSR) di DARPA sono orientati allo sviluppo di tecniche di equità dati i vincoli hardware estremi.

15.4.4 Sicurezza

I principali rischi per la sicurezza del cloud ML includono hacking dei modelli, avvelenamento dei dati e malware che interrompono i servizi cloud. Le tecniche di robustezza come l'addestramento avversario, il rilevamento delle anomalie e i modelli diversificati mirano a rafforzare il cloud ML contro gli attacchi. La ridondanza può aiutare a prevenire singoli punti di errore.

Edge ML e TinyML interagiscono con il mondo fisico, quindi l'affidabilità e la convalida della sicurezza sono fondamentali. Piattaforme di test rigorose come Foretellix generano sinteticamente scenari edge per convalidare la sicurezza. La sicurezza di TinyML è

amplificata da dispositivi autonomi con supervisione limitata. La sicurezza di TinyML spesso si basa sul coordinamento collettivo: sciami di droni mantengono la sicurezza tramite ridondanza. Anche le barriere di controllo fisiche limitano i comportamenti non sicuri dei dispositivi TinyML.
In sintesi, la sicurezza è fondamentale ma si manifesta in modo diverso in ogni dominio. Cloud ML protegge dall'hacking, edge ML interagisce fisicamente, quindi l'affidabilità è fondamentale e TinyML sfrutta il coordinamento distribuito per la sicurezza. La comprensione delle sfumature guida le tecniche di sicurezza appropriate.

15.4.5 Responsabilità

La responsabilità di Cloud ML si concentra su pratiche aziendali come comitati AI responsabili, carte etiche e processi per affrontare incidenti dannosi. Audit di terze parti e supervisione governativa esterna promuovono la responsabilità di Cloud ML.
La responsabilità di Edge ML è più complessa con dispositivi distribuiti e frammentazione della supply chain. Le aziende sono responsabili dei dispositivi, ma i componenti provengono da vari fornitori. Gli standard di settore aiutano a coordinare la responsabilità di Edge ML tra le parti interessate.
Con TinyML, i meccanismi di responsabilità devono essere tracciati attraverso lunghe e complesse supply chain di circuiti integrati, sensori e altro hardware. Gli schemi di certificazione TinyML aiutano a tracciare la provenienza dei componenti. Le associazioni di categoria dovrebbero idealmente promuovere la responsabilità condivisa per TinyML etico.

15.4.6 Governance

Le organizzazioni istituiscono una governance interna per il cloud ML, come comitati etici, audit e gestione del rischio del modello. Ma anche la governance esterna supervisiona il cloud ML, come le normative su pregiudizi e trasparenza come AI Bill of Rights, General Data Protection Regulation (GDPR) e California Consumer Protection Act (CCPA). L'audit di terze parti supporta la governance del cloud ML.
Edge ML è più decentralizzato e richiede un'autogovernance responsabile da parte di sviluppatori e aziende che distribuiscono modelli localmente. Le associazioni di settore coordinano la governance tra i fornitori di edge ML e il software aperto aiuta ad allineare gli incentivi per l'edge ML etico.
L'estrema decentralizzazione e complessità rendono la governance esterna impraticabile con TinyML. TinyML si basa su protocolli e standard per l'autogovernance integrati nella progettazione del modello e nell'hardware. La crittografia consente l'affidabilità dimostrabile dei dispositivi TinyML.

15.4.7 Privacy

Per il cloud ML, grandi quantità di dati utente sono concentrate nel cloud, creando rischi di esposizione tramite violazioni. Le tecniche di privacy differenziali aggiungono rumore ai dati cloud per preservare la privacy. Rigidi controlli di accesso e crittografia proteggono i dati cloud a riposo e in transito.
Edge ML sposta l'elaborazione dei dati sui dispositivi utente, riducendo la raccolta di dati aggregati ma aumentando la potenziale sensibilità poiché i dati personali risiedono sul dispositivo. Apple utilizza ML on-device e privacy differenziale per addestrare modelli riducendo al minimo la condivisione dei dati. L'anonimizzazione dei dati e le enclave sicure proteggono i dati on-device.
TinyML distribuisce i dati su molti dispositivi con risorse limitate, rendendo improbabili le violazioni centralizzate e rendendo difficile l'anonimizzazione su larga scala. La minimizzazione dei dati e l'utilizzo di dispositivi edge come intermediari aiutano la privacy di TinyML.

Quindi, mentre il cloud ML deve proteggere dati centralizzati espansivi, l'edge ML protegge i dati sensibili on-device e TinyML mira a una condivisione minima dei dati distribuiti a causa dei vincoli. Mentre la privacy è fondamentale in tutto, le tecniche devono adattarsi all'ambiente. La comprensione delle sfumature consente di selezionare approcci appropriati per la tutela della privacy.

15.5 Aspetti Tecnici

15.5.1 Rilevamento e Mitigazione dei Pregiudizi

Un'ampia mole di lavoro ha dimostrato che i modelli di apprendimento automatico possono presentare pregiudizi, da persone con prestazioni inferiori di una certa identità a decisioni che limitano l'accesso dei gruppi a risorse importanti (Buolamwini e Gebru 2018). Garantire un trattamento equo e giusto per tutti i gruppi interessati dai sistemi di apprendimento automatico è fondamentale, poiché questi modelli hanno un impatto sempre maggiore sulla vita delle persone in settori come prestiti, assistenza sanitaria e giustizia penale. In genere valutiamo l'equità del modello considerando "attributi di sottogruppo" non correlati all'attività di previsione che catturano identità come razza, genere o religione. Ad esempio, in un modello di previsione di inadempienza del prestito, i sottogruppi potrebbero includere razza, genere o religione. Quando i modelli vengono addestrati ingenuamente per massimizzare l'accuratezza, spesso ignorano le prestazioni del sottogruppo. Tuttavia, ciò può avere un impatto negativo sulle comunità emarginate.

Per illustrare, si immagini un modello che prevede il rimborso del prestito in cui i più (+) rappresentano il rimborso e i cerchi (O) rappresentano l'inadempienza, come mostrato in Figura 15.1. L'accuratezza ottimale sarebbe classificare correttamente tutto il Gruppo A, classificando erroneamente alcuni dei richiedenti affidabili del Gruppo B come inadempienti. Se le classificazioni positive consentono l'accesso ai prestiti, il Gruppo A riceverebbe molti più prestiti, il che si tradurrebbe naturalmente in un risultato distorto.

Figura 15.1: Equità e accuratezza.

In alternativa, correggere i pregiudizi contro il Gruppo B aumenterebbe probabilmente i "falsi positivi" e ridurrebbe l'accuratezza per il Gruppo A. Oppure, potremmo addestrare modelli separati focalizzati sulla massimizzazione dei veri positivi per ciascun gruppo. Tuttavia, ciò richiederebbe l'utilizzo esplicito di attributi sensibili come la razza nel processo decisionale. Come vediamo, ci sono tensioni intrinseche attorno a priorità come l'accuratezza rispetto all'equità del sottogruppo e se tenere conto esplicitamente delle classi protette. Le persone ragionevoli possono non essere d'accordo sui compromessi appropriati. I vincoli sui costi e le opzioni di implementazione complicano ulteriormente le cose. Nel complesso, garantire

l'uso equo ed etico dell'apprendimento automatico implica la navigazione di queste sfide complesse.
Pertanto, la letteratura sull'equità ha proposto tre principali *metriche di equità* per quantificare quanto sia equo un modello su un set di dati (Hardt, Price, e Srebro 2016). Dato un modello h e un set di dati D costituito da campioni (x,y,s), dove x sono le caratteristiche dei dati, y è l'etichetta e s è l'attributo del sottogruppo, e supponiamo che ci siano semplicemente due sottogruppi a e b, possiamo definire quanto segue.

1. **Parità Demografica** chiede quanto è accurato un modello per ogni sottogruppo. In altre parole, P(h(X) = Y S = a) = P(h(X) = Y S = b)
2. **Quote Equalizzate** chiede quanto è preciso un modello su campioni positivi e negativi per ogni sottogruppo. P(h(X) = y S = a, Y = y) = P(h(X) = y S = b, Y = y)
3. **Uguaglianza di Opportunità** è un caso speciale di probabilità equalizzate che chiede solo quanto è preciso un modello su campioni positivi. Ciò è rilevante in casi come l'allocazione delle risorse, in cui ci preoccupiamo di come le etichette positive (vale a dire, allocate in base alle risorse) siano distribuite tra i gruppi. Ad esempio, ci preoccupiamo che una proporzione uguale di prestiti venga concessa sia agli uomini che alle donne. P(h(X) = 1 S = a, Y = 1) = P(h(X) = 1 S = b, Y = 1)

Nota: Queste definizioni spesso adottano una visione ristretta quando si considerano confronti binari tra due sottogruppi. Un altro filone di ricerca di apprendimento automatico equo incentrato su *multi-calibrazione* e *multi-accuratezza* considera le interazioni tra un numero arbitrario di identità, riconoscendo l'intersezionalità intrinseca delle identità individuali nel mondo reale (Hébert-Johnson et al. 2018).

15.5.1.1 Il Contesto è Importante

Prima di prendere qualsiasi decisione tecnica per sviluppare un algoritmo ML imparziale, dobbiamo comprendere il contesto che circonda il nostro modello. Ecco alcune delle domande chiave su cui riflettere:

- Per chi prenderà decisioni questo modello?
- Chi è rappresentato nei dati di training?
- Chi è rappresentato e chi manca al tavolo di ingegneri, progettisti e manager?
- Che tipo di impatti duraturi potrebbe avere questo modello? Ad esempio, avrà un impatto sulla sicurezza finanziaria di un individuo su scala generazionale, come la determinazione delle ammissioni al college o l'ammissione di un prestito per una casa?
- Quali pregiudizi storici e sistematici sono presenti in questo contesto e sono presenti nei dati di training da cui il modello generalizzerà?

Comprendere il background sociale, etico e storico di un sistema è fondamentale per prevenire danni e dovrebbe informare le decisioni durante tutto il ciclo di sviluppo del modello. Dopo aver compreso il contesto, si possono prendere varie decisioni tecniche per rimuovere i pregiudizi. Innanzitutto, si deve decidere quale metrica di equità è il criterio più appropriato per l'ottimizzazione. Successivamente, ci sono generalmente tre aree principali in cui si può intervenire per eliminare i pregiudizi di un sistema ML.
Innanzitutto, la preelaborazione è quando si bilancia un set di dati per garantire una rappresentazione equa o addirittura si aumenta il peso su determinati gruppi sottorappresentati per garantire che il modello funzioni bene. In secondo luogo, nell'elaborazione si tenta di modificare il processo di training di un sistema ML per garantire che dia priorità all'equità. Questo può essere semplice come aggiungere un regolarizzatore di equità (Lowy et al. 2021) al training di un insieme di modelli e campionarli in un modo specifico (Agarwal et al. 2018).

Infine, la post-elaborazione degrada un modello dopo il fatto, prendendo un modello addestrato e modificandone le previsioni in un modo specifico per garantire che l'equità venga preservata (Alghamdi et al. 2022; Hardt, Price, e Srebro 2016). La post-elaborazione si basa sulle fasi di pre-elaborazione e in-elaborazione offrendo un'altra opportunità per affrontare i problemi di bias [pregiudizi] e equità nel modello dopo che è già stato addestrato. Il processo in tre fasi di pre-elaborazione, in-elaborazione e post-elaborazione fornisce un framework per intervenire in diverse fasi dello sviluppo del modello per mitigare i problemi relativi a pregiudizi ed equità. Mentre la pre-elaborazione e l'in-elaborazione si concentrano sui dati e sul training, la post-elaborazione consente di apportare modifiche dopo che il modello è stato completamente formato. Insieme, questi tre approcci offrono molteplici opportunità per rilevare e rimuovere pregiudizi ingiusti.

15.5.1.2 Distribuzione Ponderata

L'ampiezza delle definizioni di equità e degli interventi di debiasing esistenti sottolinea la necessità di una valutazione ponderata prima di distribuire sistemi ML. Come ricercatori e sviluppatori ML, lo sviluppo responsabile del modello richiede di istruirci in modo proattivo sul contesto del mondo reale, consultare esperti del settore e utenti finali e concentrarci sulla prevenzione dei danni.

Invece di vedere le considerazioni sull'equità come una casella da spuntare, dobbiamo impegnarci profondamente con le implicazioni sociali uniche e i compromessi etici attorno a ogni modello che costruiamo. Ogni scelta tecnica su set di dati, architetture di modelli, metriche di valutazione e vincoli di distribuzione incorpora valori. Ampliando la nostra prospettiva oltre le metriche tecniche ristrette, valutando attentamente i compromessi e ascoltando le voci interessate, possiamo lavorare per garantire che i nostri sistemi espandano le opportunità anziché codificare i pregiudizi.

La strada da seguire non risiede in una checklist di "debiasing" arbitraria, ma nell'impegno a comprendere e sostenere la nostra responsabilità etica a ogni passo. Questo impegno inizia con l'educazione proattiva di noi stessi e la consultazione degli altri, piuttosto che limitarci a seguire i movimenti di una checklist di equità. Richiede un profondo impegno nei compromessi etici nelle nostre scelte tecniche, la valutazione degli impatti su diversi gruppi e l'ascolto delle voci maggiormente interessate.

In definitiva, i sistemi di intelligenza artificiale responsabili ed etici non derivano dal "debiasing" delle caselle di controllo, ma dal rispetto del nostro dovere di valutare i danni, ampliare le prospettive, comprendere i compromessi e garantire di offrire opportunità a tutti i gruppi. Questa responsabilità etica dovrebbe guidare ogni passo.

Il collegamento tra i paragrafi è che il primo stabilisce la necessità di una valutazione ponderata delle questioni di equità piuttosto che di un approccio basato su caselle di controllo. Il secondo paragrafo si sofferma poi su come si presenta in pratica questa valutazione ponderata, ovvero impegnarsi con i compromessi, valutare gli impatti sui gruppi e ascoltare le voci interessate. Infine, l'ultimo paragrafo fa riferimento all'evitare una "checklist di debiasing arbitraria" e impegnarsi nella responsabilità etica attraverso la valutazione, la comprensione dei compromessi e l'offerta di opportunità.

15.5.2 Preservare la Privacy

Incidenti recenti hanno fatto luce su come i modelli di intelligenza artificiale possano memorizzare dati sensibili degli utenti in modi che violano la privacy. Ippolito et al. (2023) dimostra che i modelli linguistici tendono a memorizzare i dati di addestramento e possono persino riprodurre esempi di addestramento specifici. Questi rischi sono amplificati con sistemi ML personalizzati distribuiti in ambienti intimi come case o dispositivi indossabili. Prendiamo in considerazione uno smart speaker che usa le nostre conversazioni per migliorare la qualità del servizio per gli utenti che apprezzano tali miglioramenti. Sebbene

potenzialmente vantaggioso, questo crea anche rischi per la privacy, poiché i malintenzionati potrebbero tentare di estrarre ciò che lo speaker "ricorda". Il problema si estende oltre i modelli linguistici. Figura 15.2 mostra come i modelli di diffusione possono memorizzare e generare esempi di training individuali (Nicolas Carlini et al. 2023), dimostrando ulteriormente i potenziali rischi per la privacy associati ai sistemi di intelligenza artificiale che apprendono dai dati degli utenti.

Figura 15.2: Modelli di diffusione che memorizzano campioni dai dati di training. Fonte: Ippolito et al. (2023).

Man mano che l'intelligenza artificiale si integra sempre di più nella nostra vita quotidiana, sta diventando sempre più importante che le preoccupazioni sulla privacy e le solide misure di sicurezza per proteggere le informazioni degli utenti siano sviluppate con occhio critico. La sfida sta nel bilanciare i vantaggi dell'intelligenza artificiale personalizzata con il diritto fondamentale alla privacy.

Gli avversari possono usare queste capacità di memorizzazione e addestrare modelli per rilevare se specifici dati di addestramento hanno influenzato un modello target. Ad esempio, gli attacchi di inferenza di appartenenza addestrano un modello secondario che impara a rilevare un cambiamento negli output del modello target quando si effettuano inferenze sui dati su cui è stato addestrato rispetto a quelli su cui non è stato addestrato (Shokri et al. 2017).

I dispositivi ML sono particolarmente vulnerabili perché sono spesso personalizzati sui dati degli utenti e vengono distribuiti in contesti ancora più intimi come la casa. Le tecniche di apprendimento automatico privato si sono evolute per stabilire misure di sicurezza contro gli avversari, come menzionato nel capitolo Sicurezza e Privacy per combattere questi problemi di privacy. Metodi come la privacy differenziale aggiungono rumore matematico durante l'addestramento per oscurare l'influenza dei singoli punti dati sul modello. Tecniche popolari come DP-SGD (Abadi et al. 2016) tagliano anche i gradienti per limitare ciò che il modello trapelerà sui dati. Tuttavia, gli utenti dovrebbero anche avere la possibilità di eliminare l'impatto dei propri dati in un secondo momento.

15.5.3 Machine Unlearning

Con dispositivi ML personalizzati per singoli utenti e poi distribuiti su edge remoti senza connettività, sorge una sfida: come possono i modelli "dimenticare" in modo reattivo i dati dopo la distribuzione? Se gli utenti richiedono che i loro dati vengano rimossi da un modello

personalizzato, la mancanza di connettività rende impossibile la riqualificazione. Pertanto, un'efficiente dimenticanza dei dati sul dispositivo è necessaria, ma pone degli ostacoli.
Gli approcci iniziali di disapprendimento hanno incontrato delle limitazioni in questo contesto. Date le limitazioni delle risorse, recuperare modelli da zero sul dispositivo per dimenticare i dati si rivela inefficiente o addirittura impossibile. La riqualificazione completa richiede anche di conservare tutti i dati di training originali sul dispositivo, il che comporta dei rischi per la sicurezza e la privacy. Le comuni tecniche di "machine unlearning" [disapprendimento automatico] (Bourtoule et al. 2021) per sistemi ML embedded remoti non riescono a consentire la rimozione dei dati reattiva e sicura.
Tuttavia, metodi più recenti sembrano promettenti nel modificare i modelli per dimenticare approssimativamente i dati [?] senza una riqualificazione completa. Sebbene la perdita di accuratezza derivante dall'evitare ricostruzioni complete sia modesta, garantire la privacy dei dati dovrebbe comunque essere la priorità quando si gestiscono eticamente le informazioni sensibili degli utenti. Anche una minima esposizione a dati privati può violare la fiducia degli utenti. Poiché i sistemi ML diventano profondamente personalizzati, efficienza e privacy devono essere abilitate fin dall'inizio, non ripensamenti.
Le recenti discussioni sulle politiche, che includono European Union's General Data, Protection Regulation (GDPR), il California Consumer Privacy Act (CCPA), l'Act on the Protection of Personal Information (APPI) e il Consumer Privacy Protection Act (CPPA) proposto dal Canada, richiedono l'eliminazione delle informazioni private. Queste politiche, insieme a incidenti di intelligenza artificiale come la memorizzazione dei dati degli artisti da parte di Stable Diffusion, hanno sottolineato la necessità etica degli utenti di eliminare i propri dati dai modelli dopo l'addestramento.
Il diritto di rimuovere i dati nasce da preoccupazioni sulla privacy relative alle aziende o agli avversari che abusano delle informazioni sensibili degli utenti. L'unlearning automatico si riferisce alla rimozione dell'influenza di punti specifici da un modello già addestrato. Ingenuamente, ciò comporta una riqualificazione completa senza i dati eliminati. Tuttavia, i vincoli di connettività spesso rendono la riqualificazione non fattibile per i sistemi ML personalizzati e distribuiti su edge remoti. Se uno smart speaker impara da conversazioni domestiche private, è importante mantenere l'accesso per eliminare tali dati.
Sebbene limitati, i metodi si stanno evolvendo per consentire approssimazioni efficienti della riqualificazione per l'unlearning. Modificando il tempo di inferenza dei modelli, possono imitare i dati "dimenticati" senza accesso completo ai dati di addestramento. Tuttavia, la maggior parte delle tecniche attuali è limitata a modelli semplici, ha ancora costi di risorse e scambia una certa accuratezza. Sebbene i metodi si stiano evolvendo, consentire una rimozione efficiente dei dati e rispettare la privacy degli utenti rimane fondamentale per una distribuzione TinyML responsabile.

15.5.4 Esempi Avversari e Robustezza

I modelli di apprendimento automatico, in particolare le reti neurali profonde, hanno un tallone d'Achille ben documentato: spesso si rompono quando vengono apportate anche piccole perturbazioni ai loro input (Szegedy et al. 2014). Questa sorprendente fragilità evidenzia un importante divario di robustezza che minaccia l'implementazione nel mondo reale in domini ad alto rischio. Apre anche la porta ad attacchi avversari progettati per ingannare deliberatamente i modelli.
I modelli di apprendimento automatico possono mostrare una sorprendente fragilità: piccole modifiche agli input possono causare malfunzionamenti scioccanti, anche nelle reti neurali profonde all'avanguardia (Szegedy et al. 2014). Questa imprevedibilità sui dati fuori campione sottolinea le lacune nella generalizzazione e nella robustezza del modello. Data la crescente

ubiquità dell'apprendimento automatico, consente anche minacce avversarie che sfruttano i punti ciechi dei modelli.
Le reti neurali profonde dimostrano una doppia natura quasi paradossale: competenza umana nelle distribuzioni di training abbinata a un'estrema fragilità alle piccole perturbazioni di input (Szegedy et al. 2014). Questa lacuna di vulnerabilità avversaria ne evidenzia altre nelle procedure ML standard e minacce all'affidabilità nel mondo reale. Allo stesso tempo, può essere sfruttata: gli aggressori possono trovare punti di rottura del modello che gli umani non percepirebbero.
Figura 15.3 include un esempio di una piccola perturbazione insignificante che modifica una previsione del modello. Questa fragilità ha impatti nel mondo reale: la mancanza di robustezza mina la fiducia nell'implementazione di modelli per applicazioni ad alto rischio come auto a guida autonoma o diagnosi mediche. Inoltre, la vulnerabilità porta a minacce alla sicurezza: gli aggressori possono creare deliberatamente esempi avversari che sono percettivamente indistinguibili dai dati normali ma causano errori del modello.

Figura 15.3: Effetto della perturbazione sulla previsione. Fonte: Microsoft.

Ad esempio, lavori passati mostrano attacchi riusciti che ingannano i modelli per attività come il rilevamento NSFW (Bhagoji et al. 2018), il blocco degli annunci (Tramèr et al. 2019) e il riconoscimento vocale (Nicholas Carlini et al. 2016). Sebbene gli errori in questi domini rappresentino già dei rischi per la sicurezza, il problema si estende oltre la sicurezza IT. Di recente, la robustezza avversaria è stata proposta come metrica di prestazioni aggiuntiva approssimando il comportamento del caso peggiore.
La sorprendente fragilità del modello evidenziata sopra mette in dubbio l'affidabilità nel mondo reale e apre la porta alla manipolazione avversaria. Questa crescente vulnerabilità sottolinea diverse esigenze. In primo luogo, le valutazioni della robustezza morale sono essenziali per quantificare le vulnerabilità del modello prima dell'implementazione. L'approssimazione del comportamento del caso peggiore fa emergere punti ciechi.
In secondo luogo, devono essere sviluppate difese efficaci in tutti i domini per colmare queste lacune di robustezza. Con la sicurezza in gioco, gli sviluppatori non possono ignorare la minaccia di attacchi che sfruttano le debolezze del modello. Inoltre, non possiamo permetterci guasti indotti dalla fragilità per applicazioni critiche per la sicurezza come veicoli a guida autonoma e diagnosi mediche. Sono in gioco delle vite.
Infine, la comunità di ricerca continua a mobilitarsi rapidamente in risposta. L'interesse per l'apprendimento automatico avversario è esploso poiché gli attacchi rivelano la necessità di colmare il divario di robustezza tra dati sintetici e dati del mondo reale. Le conferenze ora comunemente presentano difese per proteggere e stabilizzare i modelli. La comunità riconosce che la fragilità del modello è un problema critico che deve essere affrontato tramite test di robustezza, sviluppo di difese e ricerca continua. Evidenziando i punti ciechi e rispondendo con difese basate su principi, possiamo lavorare per garantire affidabilità e sicurezza per i sistemi di apprendimento automatico, specialmente in domini ad alto rischio.

15.5.5 Creazione di Modelli Interpretabili

Poiché i modelli vengono distribuiti più frequentemente in contesti ad alto rischio, professionisti, sviluppatori, utenti finali a valle e una regolamentazione crescente hanno evidenziato la necessità di spiegabilità nell'apprendimento automatico. L'obiettivo di molti metodi di interpretabilità e spiegabilità è fornire ai professionisti maggiori informazioni sul comportamento complessivo dei modelli o sul comportamento dato un input specifico. Ciò consente agli utenti di decidere se l'output o la previsione di un modello sono affidabili o meno.

Tale analisi può aiutare gli sviluppatori a eseguire il debug dei modelli e migliorare le prestazioni evidenziando distorsioni, correlazioni spurie e modalità di errore dei modelli. Nei casi in cui i modelli possono superare le prestazioni umane in un'attività, l'interpretabilità può aiutare utenti e ricercatori a comprendere meglio le relazioni nei loro dati e pattern precedentemente sconosciuti.

Esistono molte classi di metodi di spiegabilità/interpretabilità, tra cui la spiegabilità post hoc, l'interpretabilità intrinseca e l'interpretabilità meccanicistica. Questi metodi mirano a rendere più comprensibili i modelli di apprendimento automatico complessi e a garantire che gli utenti possano fidarsi delle previsioni del modello, soprattutto in contesti critici. Fornendo trasparenza nel comportamento del modello, le tecniche di spiegabilità sono uno strumento importante per sviluppare sistemi di intelligenza artificiale sicuri, equi e affidabili.

15.5.5.1 Spiegabilità Post Hoc

I metodi di spiegabilità "post hoc" in genere spiegano il comportamento di output di un modello black-box su un input specifico. metodi più diffusi includono spiegazioni controfattuali, metodi di attribuzione delle caratteristiche e spiegazioni basate sui concetti.

Spiegazioni controfattuali, spesso chiamate anche ricorso algoritmico, "Se X non si fosse verificato, Y non si sarebbe verificato" (Wachter, Mittelstadt, e Russell 2017). Ad esempio, si consideri una persona che richiede un prestito bancario la cui richiesta viene respinta da un modello. Potrebbe chiedere alla propria banca un ricorso o come modificare per essere idonea a un prestito. Una spiegazione controfattuale indicherebbe loro quali caratteristiche devono modificare e di quanto, in modo che la previsione del modello cambi.

I metodi di attribuzione delle caratteristiche evidenziano le caratteristiche di input che sono importanti o necessarie per una particolare previsione. Per un modello di visione artificiale, ciò significherebbe evidenziare i singoli pixel che hanno contribuito maggiormente all'etichetta prevista dell'immagine. Si noti che questi metodi non spiegano in che modo quei pixel/caratteristiche influenzano la previsione, ma solo che lo fanno. I metodi comuni includono gradienti di input, GradCAM (Selvaraju et al. 2017), SmoothGrad (Smilkov et al. 2017), LIME (Ribeiro, Singh, e Guestrin 2016) e SHAP (Lundberg e Lee 2017).

Fornendo esempi di modifiche alle caratteristiche di input che altererebbero una previsione (controfattuali) o indicando le caratteristiche più influenti per una data previsione (attribuzione), queste tecniche di spiegazione post hoc fanno luce sul comportamento del modello per input individuali. Questa trasparenza granulare aiuta gli utenti a determinare se possono fidarsi e agire su output di modelli specifici.

Le spiegazioni basate sui concetti mirano a spiegare il comportamento del modello e gli output utilizzando un set predefinito di concetti semantici (ad esempio, il modello riconosce la classe di scena "camera da letto" in base alla presenza dei concetti "letto" e "cuscino"). Lavori recenti mostrano che gli utenti spesso preferiscono queste spiegazioni a quelle basate sull'attribuzione e sugli esempi perché "assomigliano al ragionamento e alle spiegazioni umane" (Vikram V. Ramaswamy et al. 2023b). I metodi di spiegazione basati sui concetti più

diffusi includono TCAV (C. J. Cai et al. 2019), Network Dissection (Bau et al. 2017) e decomposizione della base interpretabile (B. Zhou et al. 2018).
Si noti che questi metodi sono estremamente sensibili alla dimensione e alla qualità del set di concetti e c'è un compromesso tra la loro accuratezza e fedeltà e la loro interpretabilità o comprensibilità per gli esseri umani (Vikram V. Ramaswamy et al. 2023a). Tuttavia, mappando le previsioni del modello su concetti comprensibili per gli esseri umani, le spiegazioni basate sui concetti possono fornire trasparenza nel ragionamento alla base degli output del modello.

15.5.5.2 Interpretabilità Intrinseca

I modelli intrinsecamente interpretabili sono costruiti in modo tale che le loro spiegazioni siano parte dell'architettura del modello e siano quindi naturalmente fedeli, il che a volte li rende preferibili alle spiegazioni post-hoc applicate ai modelli black-box, specialmente in domini ad alto rischio in cui la trasparenza è fondamentale (Rudin 2019). Spesso, questi modelli sono vincolati in modo che le relazioni tra le caratteristiche di input e le previsioni siano facili da seguire per gli esseri umani (modelli lineari, alberi decisionali, set di decisioni, modelli k-NN) o obbediscano alla conoscenza strutturale del dominio, come la monotonicità (Maya Gupta et al. 2016), la causalità o l'additività (Lou et al. 2013; Beck e Jackman 1998). Tuttavia, lavori più recenti hanno allentato le restrizioni sui modelli intrinsecamente interpretabili, utilizzando modelli black-box per l'estrazione delle caratteristiche e un modello intrinsecamente interpretabile più semplice per la classificazione, consentendo spiegazioni fedeli che collegano le caratteristiche di alto livello alla previsione. Ad esempio, i Concept Bottleneck Models (Koh et al. 2020) prevedono un set di concetti c che viene passato in un classificatore lineare. I ProtoPNets (C. Chen et al. 2019) sezionano gli input in combinazioni lineari di somiglianze con parti prototipiche del set di training.

15.5.5.3 Interpretabilità Meccanicistica

I metodi di interpretabilità meccanicistica cercano di effettuare il reverse engineering delle reti neurali, spesso paragonandoli a come si potrebbe effettuare quello di un binario compilato o a come i neuroscienziati tentano di decodificare la funzione di singoli neuroni e circuiti nel cervello. La maggior parte delle ricerche sull'interpretabilità meccanicistica vede i modelli come un grafo computazionale (Geiger et al. 2021) e i circuiti sono sottografi con funzionalità distinte (L. Wang e Zhan 2019b). Gli attuali approcci all'estrazione di circuiti dalle reti neurali e alla comprensione della loro funzionalità si basano sull'ispezione manuale umana delle visualizzazioni prodotte dai circuiti (Olah et al. 2020).
In alternativa, alcuni approcci creano autoencoder sparsi che incoraggiano i neuroni a codificare caratteristiche interpretabili districate (Davarzani et al. 2023). Questo campo è molto più nuovo rispetto alle aree esistenti in spiegabilità e interpretabilità e, in quanto tale, la maggior parte dei lavori è generalmente esplorativa piuttosto che orientata alla soluzione. Ci sono molti problemi nell'interpretabilità meccanicistica, tra cui la polisemanticità di neuroni e circuiti, l'inconveniente e la soggettività dell'etichettatura umana e lo spazio di ricerca esponenziale per l'identificazione dei circuiti in grandi modelli con miliardi o trilioni di neuroni.

15.5.5.4 Sfide e Considerazioni

Man mano che i metodi per interpretare e spiegare i modelli progrediscono, è importante notare che gli esseri umani si fidano troppo e abusano degli strumenti di interpretabilità (Kaur et al. 2020) e che la fiducia di un utente in un modello dovuta a una spiegazione può essere indipendente dalla correttezza delle spiegazioni (Lakkaraju e Bastani 2020). Pertanto, è necessario che oltre a valutare la fedeltà/correttezza delle spiegazioni, i ricercatori debbano anche garantire che i metodi di interpretabilità siano sviluppati e implementati tenendo a

mente un utente specifico e che vengano eseguiti studi sugli utenti per valutarne l'efficacia e l'utilità nella pratica.
Inoltre, le spiegazioni devono essere adattate alle competenze dell'utente, all'attività per cui stanno utilizzando la spiegazione e alla corrispondente quantità minima di informazioni richieste affinché la spiegazione sia utile per prevenire il sovraccarico di informazioni.
Mentre interpretabilità/spiegabilità sono aree popolari nella ricerca sull'apprendimento automatico, pochissimi lavori studiano la loro intersezione con TinyML ed edge computing. Dato che un'applicazione significativa di TinyML è l'assistenza sanitaria, che spesso richiede elevata trasparenza e interpretabilità, le tecniche esistenti devono essere testate per scalabilità ed efficienza relativamente ai dispositivi edge. Molti metodi si basano su passaggi aggiuntivi "forward" e "backward" e alcuni richiedono persino un training approfondito nei modelli proxy, che non sono fattibili su microcontrollori con risorse limitate.
Detto questo, i metodi di spiegabilità possono essere molto utili nello sviluppo di modelli per dispositivi edge, in quanto possono fornire informazioni su come i dati di input e i modelli possono essere compressi e su come le rappresentazioni possono cambiare dopo la compressione. Inoltre, molti modelli interpretabili sono spesso più piccoli delle loro controparti black-box, il che potrebbe essere utile per le applicazioni TinyML.

15.5.6 Monitoraggio delle Prestazioni del Modello

Mentre gli sviluppatori possono addestrare modelli che sembrano avversarialmente robusti, equi e interpretabili prima della distribuzione, è fondamentale che sia gli utenti sia i proprietari del modello ne continuino a monitorare le prestazioni e l'affidabilità durante l'intero ciclo di vita. I dati cambiano frequentemente nella pratica, il che può spesso comportare cambiamenti nella distribuzione. Questi cambiamenti nella distribuzione possono avere un impatto profondo sulle prestazioni predittive "vanilla" del modello e sulla sua affidabilità (equità, robustezza e interpretabilità) nei dati del mondo reale.
Inoltre, le definizioni di equità cambiano frequentemente nel tempo, come ciò che la società considera un attributo protetto, e anche le competenze degli utenti che chiedono spiegazioni possono cambiare.
Per garantire che i modelli rimangano aggiornati con tali cambiamenti nel mondo reale, gli sviluppatori devono valutare continuamente i loro modelli su dati e standard attuali e rappresentativi e aggiornare i modelli quando necessario.

15.6 Sfide di Implementazione

15.6.1 Strutture Organizzative e Culturali

Sebbene innovazione e regolamentazione siano spesso viste come interessi contrapposti, molti paesi hanno ritenuto necessario fornire supervisione man mano che i sistemi di intelligenza artificiale si espandono in più settori. Come mostrato in Figura 15.4, questa supervisione è diventata cruciale poiché questi sistemi continuano a permeare vari settori e ad avere un impatto sulla vita delle persone (vedere AI incentrata sull'uomo, Capitolo 8 "Interventi e normative governative".

Figura 15.4: Come vari gruppi influenzano l'AI incentrata sull'uomo. Fonte: Shneiderman (2020).

Tra questi vi sono:

- Responsible Use of Artificial Intelligence del Canada
- General Data Protection Regulation (GDPR) dell'Unione Europea
- White Paper on Artificial Intelligence: a European approach to excellence and trust della Commissione Europea
- L'Information Commissioner's Office del Regno Unito e la Consultation on Explaining AI Decisions Guidance dell'Alan Turing Institute hanno co-badgetato la guida da parte degli individui interessati da esse.

15.6.2 Ottenere Dati di Qualità e Rappresentativi

Come discusso nel capitolo Data Engineering, la progettazione responsabile dell'IA deve avvenire in tutte le fasi della pipeline, inclusa la raccolta dei dati. Ciò solleva la domanda: cosa significa che i dati siano di alta qualità e rappresentativi? Consideriamo i seguenti scenari che *ostacolano* la rappresentatività dei dati:

15.6.2.1 Squilibrio dei Sottogruppi

Questo è probabilmente ciò che viene in mente quando si sente parlare di "dati rappresentativi". Lo squilibrio dei sottogruppi significa che il set di dati contiene relativamente più dati da un sottogruppo rispetto a un altro. Questo squilibrio può influire negativamente sul modello ML a valle, facendolo sovradimensionare per un sottogruppo di persone e con prestazioni scadenti per un altro.

Un esempio di conseguenza dello squilibrio dei sottogruppi è la discriminazione razziale nella tecnologia di riconoscimento facciale (Buolamwini e Gebru 2018); gli algoritmi commerciali di riconoscimento facciale hanno tassi di errore fino al 34% peggiori sulle donne dalla pelle scura rispetto agli uomini dalla pelle chiara.

Si noti che lo squilibrio dei dati è reciproco e i sottogruppi possono anche essere *sovrarappresentati* in modo dannoso nel set di dati. Ad esempio, l'Allegheny Family Screening Tool (AFST) prevede la probabilità che un bambino venga alla fine allontanato da una casa. L'AFST produce punteggi sproporzionati per diversi sottogruppi, uno dei motivi è che è basato su dati storicamente distorti, provenienti da sistemi legali penali minorili e per adulti, agenzie di assistenza pubblica e agenzie e programmi di salute comportamentale.

15.6.2.2 Quantificazione dei Risultati Target

Ciò si verifica in applicazioni in cui l'etichetta di verità di base non può essere misurata o è difficile da rappresentare in una singola quantità. Ad esempio, un modello ML in un'applicazione mobile per il benessere potrebbe voler prevedere i livelli di stress individuali. Le vere etichette di stress sono impossibili da ottenere direttamente e devono essere dedotte da altri segnali biologici, come la variabilità della frequenza cardiaca e i dati auto-riportati dall'utente. In queste situazioni, il rumore è incorporato nei dati per progettazione, rendendo questo un compito ML impegnativo.

15.6.2.3 Spostamento della Distribuzione

I dati potrebbero non rappresentare più un compito se un evento esterno importante causa un drastico cambiamento della fonte dati. Il modo più comune di pensare alle "distribution shift" [spostamenti della distribuzione] è rispetto al tempo; ad esempio, i dati sulle abitudini di acquisto dei consumatori raccolti prima del Covid potrebbero non essere più presenti nel comportamento dei consumatori oggi.

Il trasferimento provoca un'altra forma di spostamento della distribuzione. Ad esempio, quando si applica un sistema di triage addestrato sui dati di un ospedale a un altro, potrebbe verificarsi uno spostamento nella distribuzione se i due ospedali sono molto diversi.

15.6.2.4 Raccolta Dati

Una soluzione ragionevole per molti dei problemi di cui sopra con dati non rappresentativi o di bassa qualità è raccoglierne di più; possiamo raccogliere più dati mirati a un sottogruppo sottorappresentato o dall'ospedale target a cui il nostro modello potrebbe essere trasferito. Tuttavia, per alcune ragioni, raccogliere più dati è una soluzione inappropriata o non fattibile per il compito da svolgere.

- *La raccolta dati può essere dannosa*. Questo è il *paradosso dell'esposizione*, la situazione in cui coloro che traggono un guadagno significativo dalla raccolta dei propri dati sono anche coloro che sono messi a rischio dal processo di raccolta (D'ignazio e Klein (2023), Capitolo 4). Ad esempio, raccogliere più dati su individui non binari può essere importante per garantire l'equità dell'applicazione ML, ma li espone anche a rischi, a seconda di chi raccoglie i dati e di come (se i dati sono facilmente identificabili, contengono contenuti sensibili, ecc.).
- *La raccolta dati può essere costosa*. In alcuni ambiti, come l'assistenza sanitaria, ottenere dati può essere costoso in termini di tempo e denaro.
- *Raccolta dati distorta*. Le cartelle cliniche elettroniche sono un'enorme fonte di dati per le applicazioni sanitarie basate su ML. A parte i problemi di rappresentazione dei sottogruppi, i dati stessi possono essere raccolti in modo distorto. Ad esempio, il linguaggio negativo ("non aderente", "non disposto") è utilizzato in modo sproporzionato sui pazienti neri (Himmelstein, Bates, e Zhou 2022).

Concludiamo con diverse strategie aggiuntive per mantenere la qualità dei dati. Innanzitutto, è fondamentale promuovere una comprensione più approfondita dei dati. Ciò può essere ottenuto tramite l'implementazione di etichette e misure standardizzate della qualità dei dati, come nel Data Nutrition Project. Collaborare con le organizzazioni responsabili della raccolta dei dati aiuta a garantire che i dati vengano interpretati correttamente. In secondo luogo, è importante impiegare strumenti efficaci per l'esplorazione dei dati. Le tecniche di visualizzazione e le analisi statistiche possono rivelare problemi con i dati. Infine, stabilire un ciclo di feedback all'interno della pipeline ML è essenziale per comprendere le implicazioni reali dei dati. Le metriche, come le misure di equità, ci consentono di definire la "qualità dei dati" nel contesto dell'applicazione downstream; il miglioramento dell'equità può migliorare direttamente la qualità delle previsioni che gli utenti finali ricevono.

15.6.3 Bilanciamento di Accuratezza e Altri Obiettivi

I modelli di apprendimento automatico vengono spesso valutati solo in base all'accuratezza, ma questa singola metrica non riesce a catturare completamente le prestazioni del modello e i compromessi per i sistemi di intelligenza artificiale responsabili. Altre dimensioni etiche, come correttezza, robustezza, interpretabilità e privacy, possono competere con la pura accuratezza predittiva durante lo sviluppo del modello. Ad esempio, modelli intrinsecamente interpretabili come piccoli alberi decisionali o classificatori lineari con funzionalità semplificate barattano intenzionalmente una certa accuratezza per la trasparenza nel comportamento del modello e nelle previsioni. Mentre questi modelli semplificati raggiungono una minore accuratezza non catturando tutta la complessità nel set di dati, una migliore interpretabilità crea fiducia consentendo l'analisi diretta da parte di professionisti umani.

Inoltre, alcune tecniche pensate per migliorare la robustezza avversaria, come esempi di training avversario o riduzione della dimensionalità, possono degradare l'accuratezza dei dati di convalida puliti. In applicazioni sensibili come l'assistenza sanitaria, concentrarsi strettamente sull'accuratezza all'avanguardia comporta rischi etici se consente ai modelli di fare più affidamento su correlazioni spurie che introducono distorsioni o utilizzano ragionamenti opachi. Pertanto, gli obiettivi di prestazione appropriati dipendono in larga misura dal contesto socio-tecnico.

Metodologie come Value Sensitive Design forniscono framework per valutare formalmente le priorità di vari stakeholder all'interno del sistema di distribuzione nel mondo reale. Ciò spiega le tensioni tra valori quali accuratezza, interpretabilità ed equità, che possono quindi orientare decisioni di compromesso responsabili. Per un sistema di diagnosi medica, raggiungere la massima accuratezza potrebbe non essere l'obiettivo unico: migliorare la trasparenza per creare fiducia nei professionisti o ridurre i pregiudizi verso i gruppi minoritari potrebbe giustificare piccole perdite di accuratezza. L'analisi del contesto socio-tecnico è fondamentale per stabilire questi obiettivi.

Adottando una visione olistica, possiamo bilanciare responsabilmente l'accuratezza con altri obiettivi etici per il successo del modello. Il monitoraggio continuo delle prestazioni lungo più dimensioni è fondamentale man mano che il sistema si evolve dopo la distribuzione.

15.7 Considerazioni Etiche Nella Progettazione dell'IA

Dobbiamo discutere almeno di alcune delle numerose questioni etiche in gioco nella progettazione e nell'applicazione di sistemi di intelligenza artificiale e di diversi framework per affrontare tali questioni, tra cui quelle relative alla sicurezza dell'intelligenza artificiale, all'interazione uomo-computer (HCI) e alla scienza, tecnologia e società (STS).

15.7.1 Sicurezza dell'Intelligenza Artificiale e Allineamento dei Valori

Nel 1960, Norbert Weiner scrisse: "'se utilizziamo, per raggiungere i nostri scopi, un'agenzia meccanica con il cui funzionamento non possiamo interferire efficacemente... faremmo meglio ad essere abbastanza sicuri che lo scopo attribuito alla macchina sia lo scopo che desideriamo" (Wiener 1960).

Negli ultimi anni, poiché le capacità dei modelli di deep learning hanno raggiunto, e talvolta persino superato, le capacità umane, la questione della creazione di sistemi di intelligenza artificiale che agiscano in accordo con le intenzioni umane invece di perseguire obiettivi non intenzionali o indesiderati è diventata fonte di preoccupazione (Russell 2021). Nel campo della sicurezza dell'IA, un obiettivo particolare riguarda "l'allineamento dei valori", ovvero il problema di come codificare lo scopo "giusto" nelle macchine Intelligenza artificiale compatibile con gli esseri umani. L'attuale ricerca sull'IA presuppone che conosciamo gli

obiettivi che vogliamo raggiungere e "studia la capacità di raggiungere gli obiettivi, non la progettazione di tali obiettivi".
Tuttavia, i complessi contesti di distribuzione nel mondo reale rendono difficile definire esplicitamente "lo scopo giusto" per le macchine, richiedendo quadri per l'impostazione di obiettivi responsabili ed etici. Metodologie come Value Sensitive Design forniscono meccanismi formali per far emergere le tensioni tra i valori e le priorità delle parti interessate. Adottando una visione socio-tecnica olistica, possiamo garantire meglio che i sistemi intelligenti perseguano obiettivi che si allineano con ampie intenzioni umane anziché massimizzare metriche ristrette come la sola accuratezza. Raggiungere questo obiettivo nella pratica rimane una questione di ricerca aperta e critica man mano che le capacità dell'IA avanzano rapidamente.
L'assenza di questo allineamento può portare a diversi problemi di sicurezza dell'IA, come documentato in una varietà di modelli di deep learning. Una caratteristica comune dei sistemi che ottimizzano per un obiettivo è che le variabili non direttamente incluse nell'obiettivo possono essere impostate su valori estremi per aiutare a ottimizzare per quell'obiettivo, portando a problemi caratterizzati come gioco di specifiche, hacking di ricompensa, ecc., nel "reinforcement learning (RL)" [apprendimento per rinforzo].
Negli ultimi anni, un'implementazione particolarmente popolare di RL è stata quella dei modelli pre-addestrati utilizzando apprendimento auto-supervisionato e "Reinforcement Learning From Human Feedback (RLHF)" [apprendimento per rinforzo fine-tuned da feedback umano] (Christiano et al. 2017). Ngo 2022 (Ngo, Chan, e Mindermann 2022) sostiene che premiando i modelli per apparire innocui ed etici e massimizzando al contempo i risultati utili, RLHF potrebbe incoraggiare l'emergere di tre proprietà problematiche: hacking della ricompensa consapevole della situazione, in cui le politiche sfruttano la fallibilità umana per ottenere un'elevata ricompensa, obiettivi rappresentati internamente non allineati che si generalizzano oltre la distribuzione di messa a punto RLHF e strategie di ricerca del potere.
Allo stesso modo, Van Noorden (2016) delinea sei problemi concreti per la sicurezza dell'IA, tra cui evitare effetti collaterali negativi, evitare hacking della ricompensa, supervisione scalabile per aspetti dell'obiettivo che sono troppo costosi per essere valutati frequentemente durante il training, strategie di esplorazione sicure che incoraggiano la creatività prevenendo al contempo i danni e robustezza allo spostamento distributivo in ambienti di test invisibili.

15.7.2 Sistemi Autonomi e Controllo [e Fiducia]

Le conseguenze dei sistemi autonomi che agiscono indipendentemente dalla supervisione umana e spesso al di fuori del giudizio umano sono state ampiamente documentate in diversi settori e casi d'uso. Più di recente, il Dipartimento dei veicoli a motore della California ha sospeso i permessi di distribuzione e collaudo di Cruise per i suoi veicoli autonomi, citando "rischi irragionevoli per la sicurezza pubblica". Uno di questi incidenti si è verificato quando un veicolo ha colpito un pedone che stava attraversando le strisce pedonali dopo che il semaforo era diventato verde e al veicolo è stato permesso di procedere. Nel 2018, un pedone che attraversava la strada con la sua bicicletta è morto quando un'auto Uber a guida autonoma, che operava in modalità autonoma, non è riuscita a classificare accuratamente il suo corpo in movimento come un oggetto da evitare.
Anche i sistemi autonomi oltre ai veicoli a guida autonoma sono suscettibili a tali problemi, con conseguenze potenzialmente più gravi, poiché i droni alimentati da remoto stanno già rimodellando la guerra. Sebbene tali incidenti sollevino importanti questioni etiche su chi dovrebbe essere ritenuto responsabile quando questi sistemi falliscono, evidenziano anche le sfide tecniche nel dare il pieno controllo di attività complesse e reali alle macchine.

In sostanza, c'è una tensione tra autonomia umana e delle macchine. Le discipline ingegneristiche e informatiche hanno teso a concentrarsi sull'autonomia delle macchine. Ad esempio, a partire dal 2019, una ricerca della parola "autonomia" nella Digital Library dell'Association for Computing Machinery (ACM) rivela che dei 100 articoli più citati, il 90% riguarda l'autonomia delle macchine (Calvo et al. 2020). Nel tentativo di costruire sistemi a beneficio dell'umanità, queste discipline hanno assunto, senza dubbio, l'aumento della produttività, dell'efficienza e dell'automazione come strategie primarie per il beneficio dell'umanità.

Questi obiettivi pongono l'automazione delle macchine in prima linea, spesso a spese dell'uomo. Questo approccio soffre di sfide intrinseche, come notato fin dai primi giorni dell'IA attraverso il "Frame problem" [specifica degli effetti] e il "qualification problem" [qualificazione delle precondizioni] (cfr. http://www.diag.uniroma1.it/~nardi/Didattica/RC/lezioni/sitcalc-1.pdf), che formalizza l'osservazione che è impossibile specificare tutte le precondizioni necessarie per il successo di un'azione nel mondo reale (McCarthy 1981).

Queste limitazioni logiche hanno dato origine ad approcci matematici come la "Responsibility-sensitive safety (RSS)" [sicurezza sensibile alla responsabilità] (Shalev-Shwartz, Shammah, e Shashua 2017), che mira a scomporre l'obiettivo finale di un sistema di guida automatizzato (vale a dire la sicurezza) in condizioni concrete e verificabili che possono essere rigorosamente formulate in termini matematici. L'obiettivo dell'RSS è che tali norme di sicurezza garantiscano la sicurezza del "Automated Driving System (ADS)" [sistema di guida autonoma] nella rigorosa forma di dimostrazione matematica. Tuttavia, tali approcci tendono a utilizzare l'automazione per affrontare i problemi dell'automazione e sono suscettibili a molti degli stessi problemi.

Un altro approccio per combattere questi problemi è concentrarsi sulla progettazione "human-centered" di sistemi interattivi che incorporano il controllo umano. Il design sensibile al valore (Friedman 1996) ha descritto tre fattori di progettazione chiave per un'interfaccia utente che hanno un impatto sull'autonomia, tra cui capacità del sistema, complessità, rappresentazione errata e fluidità. Un modello più recente, chiamato METUX (A Model for Motivation, Engagement, and Thriving in the User Experience), sfrutta le intuizioni della "Self-determination Theory (SDT)" in psicologia per identificare sei sfere distinte dell'esperienza tecnologica che contribuiscono ai sistemi di progettazione che promuovono il benessere e la prosperità umana (Peters, Calvo, e Ryan 2018). SDT definisce l'autonomia come agire in base ai propri obiettivi e valori, il che è distinto dall'uso dell'autonomia come semplice sinonimo di indipendenza o di controllo (Ryan e Deci 2000).

Calvo et al. (2020) elabora METUX e le sue sei "sfere di esperienza tecnologica" nel contesto dei sistemi di raccomandazione AI. Propongono queste sfere (Adozione, Interfaccia, Attività, Comportamento, Vita e Società) come un modo per organizzare il pensiero e la valutazione della progettazione tecnologica al fine di catturare in modo appropriato gli impatti contraddittori e a valle sull'autonomia umana quando interagisce con i sistemi AI.

15.7.3 Impatti Economici su Posti di Lavoro, Competenze, Salari

Una delle principali preoccupazioni dell'attuale ascesa delle tecnologie AI è la disoccupazione diffusa. Con l'espansione delle capacità dei sistemi AI, molti temono che queste tecnologie causeranno una perdita assoluta di posti di lavoro, poiché sostituiranno i lavoratori attuali e supereranno ruoli occupazionali alternativi in tutti i settori. Tuttavia, il cambiamento dei panorami economici per mano dell'automazione non è una novità e, storicamente, si è scoperto che riflette pattern di *spostamento* piuttosto che di sostituzione (Shneiderman 2022)—Capitolo 4. In particolare, l'automazione di solito riduce i costi e

aumenta la qualità, aumentando notevolmente l'accesso e la domanda. La necessità di servire questi mercati in crescita spinge la produzione, creando nuovi posti di lavoro.
Inoltre, gli studi hanno scoperto che i tentativi di raggiungere un'automazione "lights-out", ovvero un'automazione produttiva e flessibile con un numero minimo di lavoratori umani, non hanno avuto successo. I tentativi di farlo hanno portato a quella che la task force del MIT Work of the Future ha definito "automazione a somma zero", in cui la flessibilità dei processi viene sacrificata per aumentare la produttività.
Al contrario, la task force propone un approccio di "automazione a somma positiva" in cui la flessibilità viene aumentata progettando una tecnologia che incorpora strategicamente gli esseri umani dove sono molto necessari, rendendo più facile per i dipendenti della linea addestrare e correggere i robot, utilizzando un approccio bottom-up per identificare quali attività dovrebbero essere automatizzate; e scegliendo le giuste metriche per misurare il successo (vedi Work of the Future del MIT).
Tuttavia, l'ottimismo delle prospettive di alto livello non esclude danni individuali, specialmente per coloro le cui competenze e lavori saranno resi obsoleti dall'automazione. La pressione pubblica e legislativa, così come gli sforzi di responsabilità sociale delle aziende, dovranno essere diretti alla creazione di politiche che condividano i vantaggi dell'automazione con i lavoratori e si traducano in salari minimi e benefici più elevati.

15.7.4 Comunicazione Scientifica e Alfabetizzazione IA

Un sondaggio del 1993 sulle convinzioni di 3000 adulti nordamericani sulla "macchina pensante elettronica" ha rivelato due prospettive principali del primo computer: la prospettiva dello "strumento utile dell'uomo" e la prospettiva della "macchina pensante fantastica". Gli atteggiamenti che contribuiscono alla visione della "macchina pensante fantastica" in questo e altri studi hanno rivelato una caratterizzazione dei computer come "cervelli intelligenti, più intelligenti delle persone, illimitati, veloci, misteriosi e spaventosi" (Martin 1993). Questi timori evidenziano una componente facilmente trascurata dell'IA responsabile, specialmente in mezzo alla corsa alla commercializzazione di tali tecnologie: la comunicazione scientifica che comunica accuratamente le capacità *e* le limitazioni di questi sistemi, fornendo al contempo trasparenza sui limiti della conoscenza degli esperti su questi sistemi.
Man mano che le capacità dei sistemi di IA si espandono oltre la comprensione della maggior parte delle persone, c'è una tendenza naturale a presumere i tipi di mondi apocalittici dipinti dai nostri media. Ciò è dovuto in parte all'apparente difficoltà di assimilare informazioni scientifiche, persino in culture tecnologicamente avanzate, che porta i prodotti della scienza a essere percepiti come magia, "comprensibili solo in termini di ciò che hanno fatto, non di come hanno funzionato" (Handlin 1965).
Mentre le aziende tecnologiche dovrebbero essere ritenute responsabili per aver limitato le affermazioni grandiose e non essere cadute in cicli di clamore, la ricerca che studia la comunicazione scientifica, in particolare per quanto riguarda l'intelligenza artificiale (generativa), sarà utile anche per tracciare e correggere la comprensione pubblica di queste tecnologie. Un'analisi del database accademico Scopus ha scoperto che tale ricerca è scarsa, con solo una manciata di articoli che menzionano sia "comunicazione scientifica" che "intelligenza artificiale" (Schäfer 2023).
La ricerca che espone le prospettive, i "frame" e le immagini del futuro promosse da istituzioni accademiche, aziende tecnologiche, stakeholder, enti regolatori, giornalisti, ONG e altri aiuterà anche a identificare potenziali lacune nell'alfabetizzazione AI tra gli adulti (Lindgren 2023). Una maggiore attenzione all'alfabetizzazione AI da parte di tutti gli stakeholder sarà importante per aiutare le persone le cui competenze sono rese obsolete dall'automazione AI (Ng et al. 2021).

"Ma anche coloro che non acquisiscono mai quella comprensione hanno bisogno di rassicurazioni sul fatto che esista una connessione tra gli obiettivi della scienza e il loro benessere e, soprattutto, che lo scienziato non sia un uomo completamente a parte, ma uno che condivide parte del loro valore." (Handlin, 1965)

15.8 Conclusione

Un'intelligenza artificiale responsabile è fondamentale poiché i sistemi di apprendimento automatico esercitano una crescente influenza nei settori sanitario, lavorativo, finanziario e della giustizia penale. Mentre l'intelligenza artificiale promette immensi benefici, i modelli progettati in modo sconsiderato rischiano di perpetrare danni attraverso pregiudizi, violazioni della privacy, comportamenti indesiderati e altre insidie.

Mantenere i principi di equità, spiegabilità, responsabilità, sicurezza e trasparenza consente lo sviluppo di un'intelligenza artificiale etica allineata ai valori umani. Tuttavia, l'implementazione di questi principi comporta il superamento di complesse sfide tecniche e sociali relative al rilevamento di pregiudizi nei set di dati, alla scelta di appropriati compromessi nei modelli, alla protezione di dati di training di qualità e altro ancora. Framework come la progettazione sensibile al valore guidano il bilanciamento dell'accuratezza rispetto ad altri obiettivi in base alle esigenze delle parti interessate.

Guardando al futuro, il progresso dell'intelligenza artificiale responsabile richiede una ricerca continua e l'impegno del settore. Sono necessari benchmark più standardizzati per confrontare pregiudizi e robustezza dei modelli. Man mano che il TinyML personalizzato si espande, abilitare una trasparenza efficiente e il controllo dell'utente per i dispositivi edge giustifica l'attenzione. Le strutture e le politiche di incentivazione riviste devono incoraggiare uno sviluppo deliberato ed etico prima di un'implementazione sconsiderata. L'istruzione sulla cultura dell'intelligenza artificiale e sui suoi limiti contribuirà ulteriormente alla comprensione pubblica.

I metodi responsabili sottolineano che, mentre l'apprendimento automatico offre un potenziale immenso, un'applicazione sconsiderata rischia di avere conseguenze negative. La collaborazione interdisciplinare e la progettazione incentrata sull'uomo sono essenziali affinché l'intelligenza artificiale possa promuovere un ampio beneficio sociale. Il percorso da seguire non risiede in una checklist arbitraria, ma in un impegno costante per comprendere e sostenere la nostra responsabilità etica a ogni passo. Intraprendendo un'azione coscienziosa, la comunità dell'apprendimento automatico può guidare l'intelligenza artificiale verso l'emancipazione di tutte le persone in modo equo e sicuro.

15.9 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Stiamo lavorando costantemente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

ⓘ Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- What am I building? What is the goal?
- Who is the audience?
- What are the consequences?
- Responsible Data Collection.

Video

- Video 15.1

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- *Prossimamente*.

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

16. IA Sostenibile

DALL·E 3 Prompt: Illustrazione 3D su uno sfondo chiaro di una rete IA sostenibile interconnessa con una miriade di fonti energetiche ecocompatibili. L'IA gestisce e ottimizza attivamente la sua energia da fonti come pannelli solari, turbine eoliche e dighe idroelettriche, enfatizzando l'efficienza energetica e le prestazioni. Reti neurali profonde si diffondono ovunque, ricevendo energia da queste risorse sostenibili.

Obiettivi dell'Apprendimento

- Comprendere l'impatto ambientale dell'IA, inclusi consumo energetico, emissioni di carbonio, rifiuti elettronici ed effetti sulla biodiversità.
- Informarsi su metodi e best practice per lo sviluppo di sistemi IA sostenibili
- Apprezzare l'importanza di adottare una prospettiva del ciclo di vita quando si valuta e si affronta la sostenibilità dei sistemi IA.
- Riconoscere i ruoli che vari stakeholder, come ricercatori, aziende, politici e utenti finali, svolgono nel promuovere un progresso IA responsabile e sostenibile.
- Scoprire framework, metriche e strumenti specifici per consentire uno sviluppo di IA più ecologico.
- Apprezzare i casi di studio del mondo reale come le pratiche di efficienza 4M di Google che mostrano come le organizzazioni stanno adottando misure tangibili per migliorare il record ambientale dell'IA

16.1 Introduzione

I rapidi progressi nell'intelligenza artificiale (IA) e nel machine learning (ML) [apprendimento automatico] hanno portato a molte applicazioni e ottimizzazioni utili per l'efficienza delle prestazioni. Tuttavia, la notevole crescita dell'IA ha un costo significativo ma spesso trascurato: il suo impatto ambientale. Il rapporto più recente pubblicato dall'IPCC, l'organismo internazionale che guida le valutazioni scientifiche del cambiamento climatico e dei suoi impatti, ha sottolineato l'importanza urgente di affrontare il cambiamento climatico. Senza sforzi immediati per ridurre le emissioni globali di CO_2 di almeno il 43 percento prima del 2030, supereremo il riscaldamento globale di 1,5 gradi Celsius (Winkler et al. 2022). Ciò potrebbe avviare cicli di feedback positivi, spingendo le temperature ancora più in alto.

Accanto alle questioni ambientali, le Nazioni Unite hanno riconosciuto 17 Sustainable Development Goals (SDG) [Obiettivi di sviluppo sostenibile], in cui l'IA può svolgere un ruolo importante e, viceversa, possono svolgere un ruolo importante nello sviluppo di sistemi di IA. Poiché il campo continua a espandersi, considerare la sostenibilità è fondamentale.
I sistemi di intelligenza artificiale, in particolare i grandi modelli linguistici come GPT-3 e i modelli di visione artificiale come DALL-E 2, richiedono enormi quantità di risorse computazionali per l'addestramento. Ad esempio, si stima che GPT-3 consumi 1.300 megawattora di elettricità, pari a 1.450 famiglie medie statunitensi in un mese intero (Maslej et al. 2023), o in altre parole, consuma abbastanza energia da rifornire una famiglia media statunitense per 120 anni! Questa immensa richiesta di energia deriva principalmente da data center affamati di energia con server che eseguono calcoli intensivi per addestrare queste complesse reti neurali per giorni o settimane.
Le stime attuali indicano che le emissioni di carbonio prodotte dallo sviluppo di un singolo modello di intelligenza artificiale sofisticato possono eguagliare le emissioni nell'arco di vita di cinque veicoli standard a benzina (Strubell, Ganesh, e McCallum 2019). Una parte significativa dell'elettricità attualmente consumata dai data center è generata da fonti non rinnovabili come carbone e gas naturale, con il risultato che i data center contribuiscono a circa l'1% delle emissioni totali di carbonio a livello mondiale. Ciò è paragonabile alle emissioni dell'intero settore delle compagnie aeree. Questa immensa impronta di carbonio dimostra l'urgente necessità di passare a fonti di energia rinnovabili come l'energia solare ed eolica per gestire lo sviluppo dell'intelligenza artificiale.
Inoltre, anche i sistemi di intelligenza artificiale su piccola scala distribuiti su dispositivi edge come parte di TinyML hanno impatti ambientali che non dovrebbero essere ignorati (Prakash, Stewart, et al. 2023). L'hardware specializzato richiesto per l'intelligenza artificiale ha un impatto ambientale dovuto all'estrazione e alla produzione di risorse naturali. GPU, CPU e chip come le TPU dipendono da metalli delle terre rare la cui estrazione e lavorazione generano un notevole inquinamento. Anche la produzione di questi componenti ha le sue richieste energetiche. Inoltre, la raccolta, l'archiviazione e la preelaborazione dei dati utilizzati per addestrare modelli sia su piccola che su larga scala comportano costi ambientali, esacerbando ulteriormente le implicazioni di sostenibilità dei sistemi ML.
Pertanto, mentre l'intelligenza artificiale promette innovazioni in molti campi, per sostenere il progresso è necessario affrontare le sfide della sostenibilità. L'intelligenza artificiale può continuare a progredire in modo responsabile ottimizzando l'efficienza dei modelli, esplorando hardware specializzato alternativo e fonti di energia rinnovabile per i data center e monitorando il suo impatto ambientale complessivo.

16.2 Responsabilità Sociale ed Etica

L'impatto ambientale dell'IA non è solo una questione tecnica, ma anche etica e sociale. Man mano che l'IA diventa sempre più integrata nelle nostre vite e nei nostri settori, la sua sostenibilità diventa sempre più critica.

16.2.1 Considerazioni Etiche

La portata dell'impatto ambientale dell'IA solleva profonde questioni etiche sulle responsabilità degli sviluppatori e delle aziende di IA nel ridurre al minimo le emissioni di carbonio e l'uso di energia. In quanto creatori di sistemi e tecnologie di IA che possono avere impatti globali di vasta portata, gli sviluppatori hanno l'obbligo etico di integrare consapevolmente la tutela ambientale nel loro processo di progettazione, anche se la sostenibilità avviene a scapito di alcuni guadagni di efficienza.
C'è una chiara e attuale necessità per noi di avere conversazioni aperte e oneste sui compromessi ambientali dell'IA all'inizio del ciclo di vita dello sviluppo. I ricercatori

dovrebbero sentirsi autorizzati a esprimere preoccupazioni se le priorità organizzative non sono allineate con gli obiettivi etici, come nel caso della lettera aperta per sospendere i giganteschi esperimenti di IA.
Inoltre, c'è una crescente necessità per le aziende di IA di esaminare attentamente i loro contributi al cambiamento climatico e al danno ambientale. Le grandi aziende tecnologiche sono responsabili dell'infrastruttura cloud, delle richieste di energia dei data center e dell'estrazione delle risorse necessarie per alimentare l'IA odierna. La leadership dovrebbe valutare se i valori e le politiche organizzative promuovano la sostenibilità, dalla produzione di hardware alle pipeline di training dei modelli.
Inoltre, potrebbe essere necessaria più di un'autoregolamentazione volontaria: i governi potrebbero dover introdurre nuove normative volte a standard e pratiche di intelligenza artificiale sostenibili se speriamo di frenare l'esplosione energetica prevista di modelli sempre più grandi. Le metriche segnalate come l'utilizzo del computer, l'impronta di carbonio e i parametri di riferimento dell'efficienza potrebbero responsabilizzare le organizzazioni.
Attraverso principi etici, politiche aziendali e regole pubbliche, i tecnici e le aziende di intelligenza artificiale hanno un profondo dovere nei confronti del nostro pianeta per garantire l'avanzamento responsabile e sostenibile della tecnologia in grado di trasformare radicalmente la società moderna. Dobbiamo alle generazioni future di fare le cose per bene.
Figura 16.1 delinea alcune preoccupazioni e sfide etiche che l'IA deve affrontare.

Figura 16.1: Sfide etiche nello sviluppo dell'IA. Fonte: COE

16.2.2 Sostenibilità a Lungo Termine

La massiccia espansione prevista dell'IA solleva urgenti preoccupazioni sulla sua sostenibilità a lungo termine. Poiché il software e le applicazioni di IA aumentano rapidamente in complessità e utilizzo in tutti i settori, la domanda di potenza di calcolo e infrastrutture salirà alle stelle in modo esponenziale nei prossimi anni.

Per mettere in prospettiva la portata della crescita prevista, la capacità di calcolo totale richiesta per l'addestramento dei modelli di IA ha visto un sorprendente aumento di 350.000 volte dal 2012 al 2019 (R. Schwartz et al. 2020). I ricercatori prevedono una crescita di oltre un ordine di grandezza ogni anno, man mano che vengono sviluppati assistenti IA personalizzati, tecnologia autonoma, strumenti di medicina di precisione e altro ancora. Le tendenze sono simili per i sistemi ML embedded, con una stima di 2,5 miliardi di dispositivi edge abilitati all'IA distribuiti entro il 2030.

La gestione di questo livello di espansione richiede innovazioni incentrate su software e hardware in termini di efficienza e integrazione rinnovabile da parte di ingegneri e scienziati dell'IA. Dal lato software, nuove tecniche di ottimizzazione dei modelli, distillazione, potatura, numeri a bassa precisione, condivisione delle conoscenze tra sistemi e altre aree devono diventare best practice diffuse per frenare le esigenze energetiche. Ad esempio, realizzare anche una domanda di elaborazione ridotta del 50% per raddoppio della capacità avrebbe un impatto enorme sull'energia totale.

Dal lato dell'infrastruttura hardware, a causa dei crescenti costi di trasferimento dati, archiviazione, raffreddamento e spazio, continuare con l'attuale modello di server farm centralizzato nei data center è probabilmente irrealizzabile a lungo termine (Lannelongue, Grealey, e Inouye 2021). Esplorare opzioni di elaborazione decentralizzate alternative attorno a "edge AI" su dispositivi locali o all'interno di reti di telecomunicazioni può alleviare le pressioni di ridimensionamento sui data center iper-scalabili ad alto consumo energetico. Allo stesso modo, il passaggio a fonti di energia rinnovabile ibride e a zero emissioni di carbonio che alimentano i principali data center dei provider cloud in tutto il mondo sarà essenziale.

16.2.3 IA per il Bene Ambientale

Sebbene molta attenzione sia rivolta alle sfide di sostenibilità dell'IA, queste potenti tecnologie forniscono soluzioni uniche per combattere il cambiamento climatico e guidare il progresso ambientale. Ad esempio, l'apprendimento automatico può ottimizzare continuamente le reti elettriche intelligenti per migliorare l'integrazione delle energie rinnovabili e l'efficienza della distribuzione dell'elettricità attraverso le reti (Dongxia Zhang, Han, e Deng 2018). I modelli possono acquisire lo stato in tempo reale di una rete elettrica e le previsioni meteorologiche per allocare e spostare le fonti rispondendo alla domanda e all'offerta.

Le reti neurali ottimizzate si sono anche dimostrate notevolmente efficaci nelle previsioni meteorologiche di prossima generazione (Lam et al. 2023) e nella modellazione climatica (Kurth et al. 2023). Possono analizzare rapidamente enormi volumi di dati climatici per potenziare la preparazione agli eventi estremi e la pianificazione delle risorse per uragani, inondazioni, siccità e altro ancora. I ricercatori del clima hanno raggiunto un'accuratezza all'avanguardia del percorso delle tempeste combinando simulazioni di IA con modelli numerici tradizionali.

L'intelligenza artificiale consente inoltre un migliore monitoraggio della biodiversità (Silvestro et al. 2022), della fauna selvatica (D. Schwartz et al. 2021), degli ecosistemi e della deforestazione illegale tramite droni e satelliti. Gli algoritmi di visione artificiale possono automatizzare le stime della popolazione delle specie e le valutazioni della salute dell'habitat su vaste regioni non monitorate. Queste capacità forniscono agli ambientalisti potenti

strumenti per combattere il bracconaggio (Bondi et al. 2018), ridurre i rischi di estinzione delle specie e comprendere i cambiamenti ecologici.
Investimenti mirati in applicazioni di intelligenza artificiale per la sostenibilità ambientale, la condivisione di dati intersettoriali e l'accessibilità dei modelli possono accelerare notevolmente le soluzioni a urgenti problemi ecologici. L'enfasi sull'intelligenza artificiale per il bene sociale indirizza l'innovazione in direzioni più pulite, guidando queste tecnologie che modellano il mondo verso uno sviluppo etico e responsabile.

16.2.4 Caso di Studio

I data center di Google sono fondamentali per alimentare prodotti come Search, Gmail e YouTube, utilizzati quotidianamente da miliardi di persone. Tuttavia, mantenere attive e funzionanti le vaste server farm richiede molta energia, in particolare per i sistemi di raffreddamento essenziali. Google si impegna costantemente per migliorare l'efficienza in tutte le operazioni. Tuttavia, i progressi si stavano rivelando difficili solo con i metodi tradizionali, considerando le complesse dinamiche personalizzate coinvolte. Questa sfida ha spinto una svolta nell'apprendimento automatico, producendo potenziali risparmi.
Dopo oltre un decennio di ottimizzazione della progettazione dei data center, invenzione di hardware di elaborazione a basso consumo energetico e protezione di fonti di energia rinnovabili, Google ha portato gli scienziati di DeepMind a sbloccare ulteriori progressi. Gli esperti di intelligenza artificiale hanno affrontato fattori complessi che circondano il funzionamento degli apparati di raffreddamento industriali. Apparecchiature come pompe e refrigeratori interagiscono in modo non lineare, mentre cambiano anche le condizioni meteorologiche esterne e le variabili architettoniche interne. Catturare questa complessità ha confuso le rigide formule ingegneristiche e l'intuizione umana.
Il team DeepMind ha sfruttato i dati storici estesi dei sensori di Google che descrivono temperature, consumi energetici e altri attributi come input di training. Hanno creato un sistema flessibile basato su reti neurali per modellare le relazioni e prevedere configurazioni ottimali, riducendo al minimo la "power usage effectiveness (PUE)" [efficacia dell'utilizzo di energia] (Barroso, Hölzle, e Ranganathan 2019); PUE è la misura standard per valutare l'efficienza con cui un data center utilizza l'energia, che fornisce la percentuale di energia totale consumata dalla struttura divisa per l'energia utilizzata direttamente per le operazioni di elaborazione. Quando testato in tempo reale, il sistema AI ha prodotto notevoli guadagni rispetto alle innovazioni precedenti, riducendo l'energia di raffreddamento del 40% per un calo del 15% nel PUE totale, un nuovo record del sito. Il framework generalizzabile ha appreso rapidamente le dinamiche di raffreddamento in condizioni mutevoli che le regole statiche non potevano eguagliare. Questa svolta evidenzia il ruolo crescente dell'AI nella trasformazione della tecnologia moderna e nell'abilitazione di un futuro sostenibile.

16.3 Consumo Energetico

16.3.1 Comprendere le Esigenze Energetiche

Comprendere le esigenze energetiche per il training e il funzionamento dei modelli di intelligenza artificiale è fondamentale nel campo in rapida evoluzione dell'intelligenza artificiale. Con l'intelligenza artificiale che sta entrando in uso diffuso in molti nuovi campi (Bohr e Memarzadeh 2020; Sudhakar, Sze, e Karaman 2023), si prevede che la domanda di dispositivi e data center abilitati all'intelligenza artificiale esploderà. Questa comprensione ci aiuta a capire perché l'intelligenza artificiale, in particolare il deep learning, è spesso etichettata come ad alta intensità energetica.

16.3.1.1 Requisiti Energetici per il Training dell'Intelligenza Artificiale

Il training di sistemi di intelligenza artificiale complessi come i grandi modelli di deep learning può richiedere livelli sorprendentemente elevati di potenza di calcolo, con profonde implicazioni energetiche. Consideriamo il modello linguistico all'avanguardia di OpenAI GPT-3 come un esempio lampante. Questo sistema spinge i confini della generazione di testo attraverso algoritmi formati su enormi set di dati. Tuttavia, l'energia consumata da GPT-3 per un singolo ciclo di addestramento potrebbe rivaleggiare con l'utilizzo mensile di un'intera cittadina. Negli ultimi anni, questi modelli di intelligenza artificiale generativa hanno guadagnato sempre più popolarità, portando a un numero maggiore di modelli addestrati. Oltre all'aumento del numero di modelli, aumenterà anche il numero di parametri in questi modelli. La ricerca mostra che l'aumento delle dimensioni del modello (numero di parametri), delle dimensioni del set di dati e del calcolo utilizzato per l'addestramento migliora le prestazioni in modo fluido senza segni di saturazione (Kaplan et al. 2020). Notare come, in Figura 16.2, il "test loss" diminuisce man mano che ciascuno dei 3 aumenta.

Figura 16.2: Le prestazioni migliorano con il calcolo, il set di dati e le dimensioni del modello. Fonte: Kaplan et al. (2020).

Cosa determina requisiti così immensi? Durante l'addestramento, modelli come GPT-3 apprendono le proprie capacità elaborando continuamente enormi volumi di dati per regolare i parametri interni. La capacità di elaborazione che consente i rapidi progressi dell'IA contribuisce anche all'aumento del consumo di energia, soprattutto quando i set di dati e i modelli aumentano a dismisura. GPT-3 evidenzia una traiettoria costante nel campo in cui ogni balzo nella sofisticazione dell'IA risale a una potenza di calcolo e risorse sempre più sostanziali. Il suo predecessore, GPT-2, richiedeva un addestramento 10 volte inferiore per calcolare solo 1,5 miliardi di parametri, una differenza ora ridotta da grandezze in quanto GPT-3 comprende 175 miliardi di parametri. Mantenere questa traiettoria verso un'intelligenza artificiale sempre più capace solleva sfide future in termini di fornitura di energia e infrastrutture.

16.3.1.2 Uso Operativo dell'Energia

Lo sviluppo e l'addestramento di modelli di intelligenza artificiale richiedono un'enorme quantità di dati, potenza di calcolo ed energia. Tuttavia, l'implementazione e il funzionamento di tali modelli comportano anche significativi costi ricorrenti di risorse nel tempo. I sistemi di intelligenza artificiale sono ora integrati in vari settori e applicazioni e stanno entrando nella vita quotidiana di una fascia demografica in crescita. Il loro impatto cumulativo sull'energia operativa e sulle infrastrutture potrebbe eclissare l'addestramento iniziale del modello.

Questo concetto si riflette nella domanda di hardware di addestramento e inferenza nei data center e nell'edge. L'inferenza si riferisce all'uso di un modello addestrato per fare previsioni o decisioni su dati del mondo reale. Secondo una recente analisi McKinsey, la necessità di sistemi avanzati per addestrare modelli sempre più grandi sta crescendo rapidamente.

Tuttavia, i calcoli di inferenza costituiscono già una parte dominante e crescente dei carichi di lavoro totali dell'intelligenza artificiale, come mostrato in Figura 16.3. L'esecuzione di inferenze in tempo reale con modelli addestrati, sia per la classificazione delle immagini, il riconoscimento vocale o l'analisi predittiva, richiede invariabilmente hardware di elaborazione come server e chip. Tuttavia, persino un modello che gestisce migliaia di richieste di riconoscimento facciale o query in linguaggio naturale ogni giorno è messo in ombra da piattaforme enormi come Meta. Dove l'inferenza su milioni di foto e video condivisi sui social media, i requisiti energetici dell'infrastruttura continuano a crescere!

Figura 16.3: Dimensioni del mercato per l'hardware di inferenza e training. Fonte: McKinsey.

Gli algoritmi che alimentano assistenti intelligenti abilitati all'intelligenza artificiale, magazzini automatizzati, veicoli a guida autonoma, assistenza sanitaria personalizzata e altro hanno un'impronta energetica individuale marginale. Tuttavia, la proliferazione prevista di queste tecnologie potrebbe aggiungere centinaia di milioni di endpoint che eseguono algoritmi di intelligenza artificiale ininterrottamente, causando un aumento della scala dei loro requisiti energetici collettivi. Gli attuali guadagni di efficienza hanno bisogno di aiuto per controbilanciare questa crescita netta.
Si prevede che l'intelligenza artificiale registrerà un tasso di crescita annuale del 37,3% tra il 2023 e il 2030. Tuttavia, applicando lo stesso tasso di crescita al computing operativo, entro il 2030 il fabbisogno energetico annuale dell'IA potrebbe moltiplicarsi fino a 1.000 volte. Quindi, mentre l'ottimizzazione del modello affronta un aspetto, l'innovazione responsabile deve anche considerare i costi totali del ciclo di vita su scale di distribuzione globali che erano inimmaginabili solo anni fa, ma che ora pongono sfide infrastrutturali e di sostenibilità.

16.3.2 Data Center e il Loro Impatto

Con l'aumento della domanda di servizi di IA, l'impatto dei data center sul consumo energetico dei sistemi di IA sta diventando sempre più importante. Sebbene queste strutture siano fondamentali per il progresso e la distribuzione dell'IA, contribuiscono in modo significativo al suo impatto energetico.

16.3.2.1 Scala

I data center sono i cavalli da tiro essenziali che consentono le recenti richieste di elaborazione dei sistemi di IA avanzati. Ad esempio, i principali provider come Meta gestiscono enormi data center che si estendono fino alle dimensioni di più campi da calcio, ospitando centinaia di migliaia di server ad alta capacità ottimizzati per l'elaborazione parallela e la produttività dei dati.

Queste enormi strutture forniscono l'infrastruttura per addestrare reti neurali complesse su vasti set di dati. Ad esempio, sulla base di informazioni trapelate, il modello linguistico GPT-4 di OpenAI è stato addestrato su data center Azure con oltre 25.000 GPU Nvidia A100, utilizzate ininterrottamente per oltre 90-100 giorni.
Inoltre, l'inferenza in tempo reale per applicazioni AI consumer su larga scala è resa possibile solo sfruttando le server farm all'interno dei data center. Servizi come Alexa, Siri e Google Assistant elaborano miliardi di richieste vocali al mese da parte di utenti in tutto il mondo, basandosi sul data center computing per una risposta a bassa latenza. In futuro, l'espansione di casi d'uso all'avanguardia come veicoli a guida autonoma, diagnosi di medicina di precisione e modelli di previsione climatica accurati richiederà risorse di calcolo significative da ottenere attingendo a vaste risorse di cloud computing on-demand dai data center. Alcune applicazioni emergenti, come le auto autonome, hanno rigidi vincoli di latenza e larghezza di banda. Sarà necessario collocare la potenza di calcolo a livello di data center sull'edge anziché sul cloud.
I prototipi di ricerca del MIT hanno mostrato camion e auto con hardware di bordo che eseguono l'elaborazione AI in tempo reale dei dati dei sensori equivalenti a piccoli data center (Sudhakar, Sze, e Karaman 2023). Questi innovativi "data center su ruote" dimostrano come veicoli come i camion a guida autonoma potrebbero aver bisogno di un calcolo su scala di data center embedded a bordo per ottenere una latenza di sistema di millisecondi per la navigazione, sebbene probabilmente ancora integrato dalla connettività wireless 5G a data center cloud più potenti.
La larghezza di banda, lo storage e le capacità di elaborazione richieste per abilitare questa futura tecnologia su larga scala dipenderanno in larga misura dai progressi nell'infrastruttura dei data center e dalle innovazioni algoritmiche dell'intelligenza artificiale.

16.3.2.2 Domanda di Energia

La domanda di energia dei data center può essere approssimativamente suddivisa in 4 componenti: infrastruttura, rete, storage e server. In Figura 16.4, vediamo che l'infrastruttura dati (che include raffreddamento, illuminazione e controlli) e i server utilizzano la maggior parte del budget energetico totale dei data center negli Stati Uniti (Shehabi et al. 2016).
Questa sezione suddivide la domanda di energia per i server e l'infrastruttura. Per quest'ultima, l'attenzione è rivolta ai sistemi di raffreddamento, poiché il raffreddamento è il fattore dominante nel consumo energetico nell'infrastruttura.

Figura 16.4: Consumo energetico dei data center negli Stati Uniti. Fonte: International Energy Agency (IEA).

16.3.2.2.1 Server

L'aumento del consumo energetico dei data center deriva principalmente dalla crescita esponenziale dei requisiti di elaborazione AI. Le macchine NVIDIA DGX H100 ottimizzate per il deep learning possono assorbire fino a 10,2 kW al picco. I principali provider gestiscono data center con centinaia o migliaia di questi nodi DGX ad alto consumo energetico collegati in rete per addestrare i più recenti modelli AI. Ad esempio, il supercomputer sviluppato per OpenAI è un singolo sistema con oltre 285.000 core CPU, 10.000 GPU e 400 gigabit al secondo di connettività di rete per ogni server GPU.
I calcoli intensivi necessari per l'intera flotta densamente popolata di una struttura e l'hardware di supporto comportano che i data center assorbano decine di megawatt 24 ore su 24. Nel complesso, gli algoritmi AI avanzati continuano ad aumentare il consumo energetico dei data center man mano che vengono distribuiti più nodi DGX per tenere il passo con la crescita prevista della domanda di risorse di elaborazione AI nei prossimi anni.

16.3.2.2.2 Sistemi di Raffreddamento

Per mantenere i server robusti alimentati al massimo della capacità e freschi, i data center richiedono una capacità di raffreddamento enorme per contrastare il calore prodotto da server densamente stipati, apparecchiature di rete e altro hardware che eseguono carichi di lavoro intensivi di elaborazione senza sosta. Con grandi data center che contengono migliaia di rack di server che operano a pieno regime, sono necessarie torri di raffreddamento e refrigeratori su scala industriale, che utilizzano energia pari al 30-40% dell'impronta elettrica totale del data center (Dayarathna, Wen, e Fan 2016). Di conseguenza, le aziende sono alla ricerca di metodi di raffreddamento alternativi. Ad esempio, il data center di Microsoft in Irlanda sfrutta un fiordo vicino per scambiare calore utilizzando oltre mezzo milione di galloni [1.9 milioni di litri] di acqua di mare al giorno.
Riconoscendo l'importanza del raffreddamento efficiente dal punto di vista energetico, sono state introdotte innovazioni volte a ridurre questa domanda di energia. Tecniche come il raffreddamento gratuito, che utilizza fonti di aria o acqua esterne quando le condizioni sono favorevoli, e l'uso dell'intelligenza artificiale per ottimizzare i sistemi di raffreddamento sono esempi di come il settore si adatta. Queste innovazioni riducono il consumo energetico, i costi operativi e diminuiscono l'impatto ambientale. Tuttavia, gli aumenti esponenziali della complessità del modello AI continuano a richiedere più server e hardware di accelerazione che operano a un utilizzo più elevato, il che si traduce in una maggiore generazione di calore e in un'energia sempre maggiore utilizzata esclusivamente per il raffreddamento.

16.3.2.3 L'impatto Ambientale

L'impatto ambientale dei data center non è causato solo dal consumo energetico diretto del data center stesso (Siddik, Shehabi, e Marston 2021). Il funzionamento del data center comporta la fornitura di acqua trattata al data center e lo scarico delle acque reflue dal data center. Gli impianti idrici e di trattamento delle acque reflue sono i principali consumatori di elettricità.
Oltre al consumo di elettricità, ci sono molti altri aspetti dell'impatto ambientale di questi data center. Il consumo di acqua dei data center può portare a problemi di scarsità idrica, maggiori esigenze di trattamento delle acque e adeguate infrastrutture di scarico delle acque reflue. Inoltre, le materie prime necessarie per la costruzione e la trasmissione di rete hanno un impatto considerevole sull'ambiente e i componenti nei data center devono essere aggiornati e sottoposti a manutenzione. Laddove quasi il 50 percento dei server è stato aggiornato entro 3 anni di utilizzo, i cicli di aggiornamento hanno dimostrato di rallentare (Davis et al. 2022). Tuttavia, ciò genera notevoli rifiuti elettronici, che possono essere difficili da riciclare.

16.3.3 Ottimizzazione Energetica

In definitiva, misurare e comprendere il consumo energetico dell'IA facilita l'ottimizzazione del consumo energetico.

Un modo per ridurre il consumo energetico di una data quantità di lavoro computazionale è eseguirlo su hardware più efficiente dal punto di vista energetico. Ad esempio, i chip TPU possono essere più efficienti dal punto di vista energetico rispetto alle CPU quando si tratta di eseguire grandi calcoli tensoriali per l'IA, poiché le TPU possono eseguire tali calcoli molto più velocemente senza consumare molta più energia delle CPU. Un altro modo è quello di creare sistemi software consapevoli del consumo energetico e delle caratteristiche dell'applicazione. Buoni esempi sono lavori di sistema come Zeus (J. You, Chung, e Chowdhury 2023) e Perseus (Chung et al. 2023), entrambi caratterizzati dal compromesso tra tempo di calcolo e consumo energetico a vari livelli di un sistema di addestramento ML per ottenere una riduzione energetica senza rallentamento end-to-end. In realtà, costruire sia hardware che software a basso consumo energetico e combinarne i vantaggi dovrebbe essere promettente, insieme a framework open source (ad esempio, Zeus) che facilitano gli sforzi della comunità.

16.4 Impronta di Carbonio

Le enormi richieste di elettricità dei data center possono portare a significative esternalità ambientali in assenza di un'adeguata fornitura di energia rinnovabile. Molte strutture dipendono fortemente da fonti di energia non rinnovabili come carbone e gas naturale. Ad esempio, si stima che i data center producano fino al 2% delle emissioni globali totali di CO_2, il che sta colmando il divario con il settore aereo. Come accennato nelle sezioni precedenti, le richieste di elaborazione dell'intelligenza artificiale sono destinate ad aumentare. Le emissioni di questa ondata sono triplici. In primo luogo, si prevede che i data center aumenteranno di dimensioni (Yanan Liu et al. 2020). In secondo luogo, le emissioni durante il training sono destinate ad aumentare in modo significativo (D. Patterson et al. 2022). In terzo luogo, le chiamate di inferenza a questi modelli sono destinate ad aumentare drasticamente. Senza azioni, questa crescita esponenziale della domanda rischia di aumentare ulteriormente l'impronta di carbonio dei data center a livelli insostenibili. I principali fornitori hanno promesso la neutralità carbonica e impegnato fondi per garantire energia pulita, ma i progressi rimangono incrementali rispetto ai piani di espansione complessivi del settore. Politiche di decarbonizzazione della rete più radicali e investimenti in energia rinnovabile potrebbero rivelarsi essenziali per contrastare l'impatto climatico dell'ondata imminente di nuovi data center volti a supportare la prossima generazione di IA.

16.4.1 Definizione e Significato

Il concetto di "impronta di carbonio" è emerso come una metrica chiave. Questo termine si riferisce alla quantità totale di gas serra, in particolare anidride carbonica, emessi direttamente o indirettamente da un individuo, un'organizzazione, un evento o un prodotto. Queste emissioni contribuiscono in modo significativo all'effetto serra, accelerando il riscaldamento globale e il cambiamento climatico. L'impronta di carbonio è misurata in termini di equivalenti di anidride carbonica (CO_2e), consentendo un resoconto completo che include vari gas serra e il loro relativo impatto ambientale. Esempi di ciò applicato ad attività di ML su larga scala sono mostrati in Figura 16.5.

Figura 16.5: Impronta di carbonio delle attività di ML su larga scala. Fonte: C.-J. Wu et al. (2022).

Considerare l'impronta di carbonio è particolarmente importante nel rapido progresso dell'IA e nella sua integrazione in vari settori, mettendone in evidenza l'impatto ambientale. I sistemi di IA, in particolare quelli che comportano calcoli intensivi come il deep learning e l'elaborazione di dati su larga scala, sono noti per le loro notevoli richieste di energia. Questa energia, spesso ricavata dalle reti elettriche, potrebbe ancora basarsi prevalentemente sui combustibili fossili, il che comporta significative emissioni di gas serra.
Prendiamo ad esempio l'addestramento di grandi modelli di IA come GPT-3 o complesse reti neurali. Questi processi richiedono un'immensa potenza di calcolo, in genere fornita dai data center. Il consumo energetico associato al funzionamento di questi centri, in particolare per attività ad alta intensità, comporta notevoli emissioni di gas serra. Gli studi hanno evidenziato che l'addestramento di un singolo modello di intelligenza artificiale può generare emissioni di carbonio paragonabili a quelle delle emissioni di più auto nel corso della loro vita, facendo luce sul costo ambientale dello sviluppo di tecnologie di intelligenza artificiale avanzate (Dayarathna, Wen, e Fan 2016). Figura 16.6 mostra un confronto tra le impronte di carbonio più basse e più alte, a partire da un volo di andata e ritorno tra New York e San Francisco, la vita media umana all'anno, la vita media americana all'anno, un'auto statunitense incluso il carburante nel corso della vita e un modello Transformer con ricerca di architettura neurale, che ha l'impronta più alta.

Figura 16.6: Impronta di carbonio del modello NLP in libbre di CO_2 equivalente. Fonte: Dayarathna, Wen, e Fan (2016).

Inoltre, l'impronta di carbonio dell'IA si estende oltre la fase operativa. L'intero ciclo di vita dei sistemi di IA, inclusa la produzione di hardware di elaborazione, l'energia utilizzata nei data center per il raffreddamento e la manutenzione e lo smaltimento dei rifiuti elettronici, contribuisce alla loro impronta di carbonio complessiva. Abbiamo discusso alcuni di questi aspetti in precedenza e discuteremo degli aspetti relativi ai rifiuti più avanti in questo capitolo.

16.4.2 La Necessità di Consapevolezza e Azione

Comprendere l'impronta di carbonio dei sistemi di IA è fondamentale per diversi motivi. In primo luogo, è un passo avanti verso la mitigazione degli impatti del cambiamento climatico. Man mano che l'intelligenza artificiale continua a crescere e a permeare diversi aspetti delle nostre vite, il suo contributo alle emissioni globali di carbonio diventa una preoccupazione significativa. La consapevolezza di queste emissioni può informare le decisioni prese da sviluppatori, aziende, decisori politici e persino ingegneri e scienziati ML come noi per garantire un equilibrio tra innovazione tecnologica e responsabilità ambientale.
Inoltre, questa comprensione stimola la spinta verso la 'Green AI' (R. Schwartz et al. 2020). Questo approccio si concentra sullo sviluppo di tecnologie di intelligenza artificiale efficienti, potenti e sostenibili dal punto di vista ambientale. Incoraggia l'esplorazione di algoritmi ad alta efficienza energetica, l'utilizzo di fonti di energia rinnovabili nei data center e l'adozione di pratiche che riducano l'impatto ambientale complessivo dell'intelligenza artificiale.
In sostanza, l'impronta di carbonio è una considerazione essenziale nello sviluppo e nell'applicazione delle tecnologie di intelligenza artificiale. Man mano che l'intelligenza artificiale si evolve e le sue applicazioni diventano più diffuse, la gestione della sua impronta di carbonio è fondamentale per garantire che questo progresso tecnologico sia in linea con gli obiettivi più ampi di sostenibilità ambientale.

16.4.3 Stima dell'Impronta di Carbonio dell'IA

Stimare l'impronta di carbonio dei sistemi di IA è fondamentale per comprendere il loro impatto ambientale. Ciò comporta l'analisi dei vari elementi che contribuiscono alle emissioni durante il ciclo di vita delle tecnologie di intelligenza artificiale e l'impiego di metodologie specifiche per quantificare accuratamente tali emissioni. Sono stati proposti molti metodi diversi per quantificare le emissioni di carbonio dell'apprendimento automatico.
L'impronta di carbonio dell'intelligenza artificiale comprende diversi elementi chiave, ognuno dei quali contribuisce all'impatto ambientale complessivo. Innanzitutto, l'energia viene consumata durante le fasi di addestramento e operative del modello di intelligenza artificiale.

La fonte di questa energia influenza pesantemente le emissioni di carbonio. Una volta addestrati, questi modelli, a seconda della loro applicazione e scala, continuano a consumare elettricità durante il funzionamento. Oltre alle considerazioni energetiche, anche l'hardware utilizzato stressa l'ambiente.

L'impronta di carbonio varia in modo significativo in base alle fonti di energia utilizzate. La composizione delle fonti che forniscono l'energia utilizzata nella rete varia ampiamente a seconda della regione geografica e persino del momento in un singolo giorno! Ad esempio, negli Stati Uniti, circa il 60 percento dell'approvvigionamento energetico totale è ancora coperto da combustibili fossili. Le fonti di energia nucleare e rinnovabili coprono il restante 40 percento. Queste frazioni non sono costanti durante il giorno. Poiché la produzione di energia rinnovabile solitamente si basa su fattori ambientali, come la radiazione solare e i campi di pressione, non forniscono una fonte di energia costante.

La variabilità della produzione di energia rinnovabile è stata una sfida continua nell'uso diffuso di queste fonti. Guardando Figura 16.7, che mostra i dati per la rete europea, vediamo che dovrebbe essere in grado di produrre la quantità di energia richiesta durante il giorno. Mentre l'energia solare raggiunge il picco a metà giornata, quella eolica ha due picchi distinti, al mattino e alla sera. Attualmente, facciamo affidamento su metodi di produzione di energia basati su combustibili fossili e carbone per supplire alla mancanza di energia nei periodi in cui le energie rinnovabili non soddisfano il fabbisogno.

È necessaria l'innovazione nelle soluzioni di accumulo di energia per consentire un uso costante di fonti di energia rinnovabile. Il carico energetico di base è attualmente soddisfatto dall'energia nucleare. Questa fonte energetica costante non produce direttamente emissioni di carbonio, ma deve essere più rapida per adattarsi alla variabilità delle fonti energetiche rinnovabili. Le aziende tecnologiche come Microsoft hanno mostrato interesse per le fonti di energia nucleare per alimentare i loro data center. Poiché la domanda dei data center è più costante rispetto alla domanda delle normali famiglie, l'energia nucleare potrebbe essere utilizzata come fonte energetica dominante.

Figura 16.7: Fonti di energia e capacità di generazione. Fonte: Energy Charts.

Inoltre, la produzione e lo smaltimento dell'hardware AI aumentano l'impronta di carbonio. La produzione di dispositivi informatici specializzati, come GPU e CPU, richiede molta energia e risorse. Questa fase spesso si basa su fonti energetiche che contribuiscono alle emissioni di gas serra. Il processo di produzione dell'industria elettronica è stato identificato

come una delle otto grandi catene di fornitura responsabili di oltre il 50 percento delle emissioni globali (Challenge 2021). Inoltre, lo smaltimento a fine vita di questo hardware, che può portare a rifiuti elettronici, ha anche implicazioni ambientali. Come accennato, i server hanno un ciclo di aggiornamento di circa 3-5 anni. Di questi rifiuti elettronici, attualmente solo il 17,4 percento viene raccolto e riciclato correttamente. Le emissioni di carbonio di questi rifiuti elettronici hanno mostrato un aumento di oltre il 50 percento tra il 2014 e il 2020 (Singh e Ogunseitan 2022).

Come è chiaro da quanto sopra, è necessaria un'adeguata analisi del ciclo di vita per descrivere tutti gli aspetti rilevanti delle emissioni causate dall'IA. Un altro metodo è la contabilità del carbonio, che valuta la quantità di emissioni di anidride carbonica direttamente e indirettamente associate alle operazioni di IA. Questa misura utilizza in genere equivalenti di CO_2, consentendo un modo standardizzato di segnalare e valutare le emissioni.

Esercizio 16.1: Impronta di Carbonio dell'IA

Sapevate che i modelli di IA all'avanguardia che potreste utilizzare hanno un impatto ambientale? Questo esercizio approfondirà l'"impronta di carbonio" di un sistema di IA. Imparerete come le richieste energetiche dei data center, il training dei grandi modelli di IA e persino la produzione di hardware contribuiscono alle emissioni di gas serra. Discuteremo perché è fondamentale essere consapevoli di questo impatto e impareremo metodi per stimare l'impronta di carbonio dei progetti di IA. Prepariamoci ad esplorare l'intersezione tra IA e sostenibilità ambientale!

Open in Colab

16.5 Oltre l'Impronta di Carbonio

L'attuale attenzione alla riduzione delle emissioni di carbonio e del consumo energetico dei sistemi di intelligenza artificiale affronta un aspetto cruciale della sostenibilità. Tuttavia, la produzione di semiconduttori e hardware che consentono l'intelligenza artificiale comporta anche gravi impatti ambientali che ricevono relativamente meno attenzione pubblica. Costruire e gestire un impianto di fabbricazione di semiconduttori all'avanguardia, o "fab", ha notevoli requisiti di risorse e sottoprodotti inquinanti che vanno oltre un'ampia impronta di carbonio.

Ad esempio, una fabbrica all'avanguardia che produce chip come quelli a 5 nm potrebbe richiedere fino a quattro milioni di galloni di acqua pura al giorno. Questo consumo di acqua si avvicina a ciò che una città di mezzo milione di persone richiederebbe per tutte le esigenze. L'approvvigionamento di questa risorsa pone costantemente un'enorme pressione sulle falde acquifere e sui bacini idrici locali, soprattutto nelle regioni già sottoposte a stress idrico che ospitano molti centri di produzione ad alta tecnologia.

Inoltre, oltre 250 sostanze chimiche pericolose uniche vengono utilizzate in varie fasi della produzione di semiconduttori all'interno delle fab (Mills e Le Hunte 1997). Tra queste, solventi volatili come acido solforico, acido nitrico e acido fluoridrico, insieme ad arsina, fosfina e altre sostanze altamente tossiche. Per impedire lo scarico di queste sostanze chimiche sono necessari ampi controlli di sicurezza e infrastrutture di trattamento delle acque reflue per evitare la contaminazione del suolo e rischi per le comunità circostanti. Qualsiasi manipolazione chimica impropria o fuoriuscita imprevista comporta conseguenze disastrose.

Oltre al consumo di acqua e ai rischi chimici, le operazioni di fabbricazione dipendono anche dall'approvvigionamento di metalli rari, generano tonnellate di rifiuti pericolosi e possono ostacolare la biodiversità locale. Questa sezione analizzerà questi impatti critici ma meno

discussi. Con vigilanza e investimenti nella sicurezza, i danni derivanti dalla produzione di semiconduttori possono essere contenuti pur consentendo il progresso tecnologico. Tuttavia, ignorare questi problemi esternalizzati aggraverà i danni ecologici e i rischi per la salute nel lungo periodo.

16.5.1 Utilizzo e Stress Idrico

La fabbricazione di semiconduttori è un processo che richiede un consumo di acqua incredibilmente elevato. In base a un articolo del 2009, un tipico wafer di silicio da 300 mm richiede 8.328 litri di acqua, di cui 5.678 litri sono acqua ultrapura (Cope 2009). Oggi, una tipica fabbrica può utilizzare fino a quattro milioni di galloni di acqua pura. Per gestire una struttura, si prevede che l'ultima fabbrica di TSMC in Arizona utilizzerà 8,9 milioni di galloni al giorno, ovvero quasi il 3 percento dell'attuale produzione idrica della città. Per mettere le cose in prospettiva, Intel e Quantis hanno scoperto che oltre il 97% del loro consumo diretto di acqua è attribuito alle operazioni di produzione di semiconduttori all'interno delle loro strutture di fabbricazione (Cooper et al. 2011).

Quest'acqua viene ripetutamente utilizzata per rimuovere i contaminanti nelle fasi di pulizia e funge anche da refrigerante e fluido vettore nei processi di ossidazione termica, deposizione chimica e planarizzazione chimico-meccanica. Nei mesi estivi di punta, ciò equivale approssimativamente al consumo giornaliero di acqua di una città con una popolazione di mezzo milione di persone.

Nonostante si trovi in regioni con acqua a sufficienza, l'uso intensivo può depauperare gravemente le falde acquifere e i bacini di drenaggio locali. Ad esempio, la città di Hsinchu a Taiwan ha subito affondamenti delle falde acquifere e intrusioni di acqua marina nelle falde acquifere a causa dell'eccessivo pompaggio per soddisfare le richieste di approvvigionamento idrico della fabbrica della Taiwan Semiconductor Manufacturing Company (TSMC). Nelle aree interne con scarsità d'acqua come l'Arizona, sono necessari massicci apporti di acqua per supportare le fabbriche nonostante i bacini già esistenti.

Lo scarico di acqua dalle fabbriche rischia di contaminare l'ambiente oltre all'esaurimento se non trattato correttamente. Sebbene gran parte dello scarico venga riciclato all'interno della fabbrica, i sistemi di purificazione filtrano comunque metalli, acidi e altri contaminanti che possono inquinare fiumi e laghi se non gestiti con cautela (Prakash, Callahan, et al. 2023). Questi fattori rendono essenziale la gestione dell'uso dell'acqua quando si mitigano impatti più ampi sulla sostenibilità.

16.5.2 Uso di Sostanze Chimiche Pericolose

La moderna fabbricazione di semiconduttori comporta la lavorazione di molte sostanze chimiche altamente pericolose in condizioni estreme di calore e pressione (S. Kim et al. 2018). Le principali sostanze chimiche utilizzate includono:

- **Acidi forti:** Gli acidi fluoridrico, solforico, nitrico e cloridrico corrodono rapidamente gli ossidi e altri contaminanti superficiali, ma presentano anche pericoli di tossicità. Le fab possono utilizzare migliaia di tonnellate di questi acidi all'anno e l'esposizione accidentale può essere fatale per i lavoratori.
- **Solventi:** Solventi chiave come xilene, metanolo e metilisobutilchetone (MIBK) gestiscono i fotoresist dissolvibili, ma hanno effetti negativi sulla salute come irritazione della pelle/degli occhi ed effetti narcotici se maneggiati in modo improprio. Creano anche rischi di esplosione e inquinamento atmosferico.
- **Gas tossici:** Le miscele di gas contenenti arsina (AsH_3), fosfina (PH_3), diborano (B_2H_6), germano (GeH_4), ecc., sono alcune delle sostanze chimiche più letali utilizzate nelle fasi di doping e deposizione di vapore. Esposizioni minime possono

causare avvelenamento, danni ai tessuti e persino la morte senza un trattamento rapido.

- **Composti clorurati:** Le vecchie formulazioni di planarizzazione chimico-meccanica incorporavano percloroetilene, tricloroetilene e altri solventi clorurati, che da allora sono stati vietati a causa dei loro effetti cancerogeni e dell'impatto sullo strato di ozono. Tuttavia, il loro rilascio precedente minaccia ancora le falde acquifere circostanti.

Protocolli di gestione rigorosi, dispositivi di protezione per i lavoratori, ventilazione, sistemi di filtraggio/lavaggio, serbatoi di contenimento secondari e meccanismi di smaltimento specializzati sono essenziali laddove queste sostanze chimiche vengono utilizzate per ridurre al minimo i pericoli per la salute, le esplosioni, l'aria e le fuoriuscite ambientali (Wald e Jones 1987). Ma occasionalmente si verificano ancora errori umani e guasti alle apparecchiature, evidenziando perché la riduzione delle intensità chimiche di fabbricazione è uno sforzo di sostenibilità continuo.

16.5.3 Esaurimento delle Risorse

Sebbene il silicio costituisca la base, sulla Terra c'è una scorta pressoché infinita di silicio. Infatti, il silicio è il secondo elemento più abbondante trovato nella crosta terrestre, rappresentando il 27,7% della massa totale della crosta. Solo l'ossigeno supera il silicio in abbondanza all'interno della crosta. Pertanto, il silicio non è necessario da considerare per l'esaurimento delle risorse. Tuttavia, i vari metalli e materiali speciali che consentono il processo di fabbricazione dei circuiti integrati e forniscono proprietà specifiche devono ancora essere scoperti. Mantenere le scorte di queste risorse è fondamentale, ma è minacciato dalla disponibilità finita e dalle influenze geopolitiche (Nakano 2021).

Gallio, indio e arsenico sono ingredienti vitali nella formazione di semiconduttori composti ultra-efficienti nei chip ad altissima velocità adatti per applicazioni 5G e AI (H.-W. Chen 2006). Tuttavia, questi elementi rari hanno depositi naturali relativamente scarsi che si stanno esaurendo. Lo United States Geological Survey ha inserito l'indio nella sua lista delle materie prime a rischio più critiche, stimando una fornitura globale sostenibile per meno di 15 anni alla crescita attuale della domanda (E. Davies 2011).

L'elio è richiesto in grandi volumi per le fabbriche di nuova generazione per consentire un raffreddamento preciso dei wafer durante il funzionamento. Ma la relativa rarità dell'elio e il fatto che una volta rilasciato nell'atmosfera, fuoriesce rapidamente dalla Terra rendono il mantenimento delle scorte di elio estremamente impegnativo a lungo termine (E. Davies 2011). Secondo le US National Academies, in questo mercato scarsamente scambiato si stanno già verificando notevoli aumenti dei prezzi e shock dell'offerta.

Altri rischi includono il controllo della Cina sul 90% degli elementi delle terre rare fondamentali per la produzione di materiali semiconduttori (A. R. Jha 2014). Qualsiasi problema nella catena di fornitura o controversia commerciale può portare a catastrofiche carenze di materie prime, data la mancanza di alternative attuali. Insieme alle carenze di elio, risolvere la disponibilità limitata e lo squilibrio geografico nell'accesso agli ingredienti essenziali rimane una priorità del settore per la sostenibilità.

16.5.4 Generazione di Rifiuti Pericolosi

Le fabbriche di semiconduttori generano tonnellate di rifiuti pericolosi ogni anno come sottoprodotti dei vari processi chimici (Grossman 2007). I principali flussi di rifiuti includono:

- **Rifiuti gassosi:** I sistemi di ventilazione delle fab catturano gas nocivi come arsina, fosfina e germano e li filtrano per evitare l'esposizione dei lavoratori. Tuttavia, ciò produce quantità significative di gas condensato pericoloso che necessita di un trattamento specializzato.

- **COV:** I composti organici volatili come xilene, acetone e metanolo sono ampiamente utilizzati come solventi fotoresistenti e vengono evaporati come emissioni durante la cottura, l'incisione e lo stripping. I COV pongono problemi di tossicità e richiedono sistemi di lavaggio per impedirne il rilascio.
- **Acidi esausti:** Acidi forti come acido solforico, acido fluoridrico e acido nitrico si esauriscono nelle fasi di pulizia e incisione, trasformandosi in una zuppa corrosiva e tossica che può reagire pericolosamente, rilasciando calore e fumi se mescolata.
- **Fanghi:** Il trattamento delle acque degli effluenti scaricati contiene metalli pesanti concentrati, residui acidi e contaminanti chimici. I sistemi di filtropressa separano questi fanghi pericolosi.
- **Torta di filtrazione:** I sistemi di filtrazione gassosa generano torte appiccicose di diverse tonnellate di composti assorbiti pericolosi che richiedono contenimento.

Senza adeguate procedure di movimentazione, serbatoi di stoccaggio, materiali di imballaggio e contenimento secondario, lo smaltimento improprio di uno qualsiasi di questi flussi di rifiuti può causare pericolose fuoriuscite, esplosioni e rilasci nell'ambiente. Gli enormi volumi significano che anche le fabbriche ben gestite producono tonnellate di rifiuti pericolosi anno dopo anno, che richiedono un trattamento esteso.

16.5.5 Impatti sulla Biodiversità

16.5.5.1 Interruzione e Frammentazione dell'Habitat

Le fabbriche di semiconduttori necessitano di ampie aree contigue per ospitare camere bianche, strutture di supporto, stoccaggio di sostanze chimiche, trattamento dei rifiuti e infrastrutture ausiliarie. Lo sviluppo di questi vasti spazi edificati smantella inevitabilmente gli habitat esistenti, danneggiando biomi sensibili che potrebbero aver impiegato decenni per svilupparsi. Ad esempio, la costruzione di un nuovo modulo di fabbricazione potrebbe radere al suolo gli ecosistemi forestali locali da cui specie come gufi maculati e alci dipendono per sopravvivere. La rimozione totale di tali habitat minaccia gravemente le popolazioni di animali selvatici che dipendono da quei terreni.
Inoltre, condutture, canali idrici, sistemi di scarico dell'aria e dei rifiuti, strade di accesso, torri di trasmissione e altre infrastrutture di supporto frammentano gli habitat indisturbati rimanenti. Gli animali che si spostano quotidianamente per cibo, acqua e deposizione delle uova possono vedere i loro pattern di migrazione bloccati da queste barriere umane fisiche che dividono in due i corridoi precedentemente naturali.

16.5.5.2 Disturbi della Vita Acquatica

Con le fabbriche di semiconduttori che consumano milioni di galloni di acqua ultra pura ogni giorno, accedere e scaricare tali volumi rischia di alterare l'idoneità degli ambienti acquatici circostanti che ospitano pesci, piante acquatiche, anfibi e altre specie. Se la fabbrica attinge alle falde acquifere come fonte di approvvigionamento primaria, un prelievo eccessivo a tassi insostenibili può impoverire i laghi o portare all'essiccazione dei corsi d'acqua man mano che i livelli dell'acqua scendono (E. Davies 2011).
Inoltre, lo scarico di acque reflue a temperature più elevate per raffreddare le apparecchiature di fabbricazione può modificare le condizioni del fiume a valle attraverso l'inquinamento termico. Le variazioni di temperatura oltre le soglie per cui si sono evolute le specie autoctone possono interrompere i cicli riproduttivi. L'acqua più calda contiene anche meno ossigeno disciolto, fondamentale per sostenere la vita di piante e animali acquatici (LeRoy Poff, Brinson, e Day 2002). In combinazione con tracce di contaminanti residui che sfuggono ai sistemi di filtrazione, l'acqua scaricata può trasformare cumulativamente gli ambienti rendendoli molto meno abitabili per gli organismi sensibili (Till et al. 2019).

16.5.5.3 Emissioni Chimiche e Aeree

Mentre le moderne fabbriche di semiconduttori mirano a contenere gli scarichi di aria e sostanze chimiche attraverso sistemi di filtraggio estesi, alcuni livelli di emissioni spesso persistono, aumentando i rischi per la flora e la fauna vicine. Gli inquinanti atmosferici possono essere trasportati sottovento, tra cui composti organici volatili (COV), composti di ossido di azoto (NOx), particolato proveniente da scarichi operativi delle fabbriche ed emissioni di carburante delle centrali elettriche.

Poiché i contaminanti permeano i terreni e le fonti d'acqua locali, la fauna selvatica che ingerisce cibo e acqua contaminati ingerisce sostanze tossiche, che la ricerca dimostra possono ostacolare la funzione cellulare, i tassi di riproduzione e la longevità, avvelenando lentamente gli ecosistemi (Hsu et al. 2016).

Allo stesso modo, le fuoriuscite accidentali di sostanze chimiche e la gestione impropria dei rifiuti, che rilasciano acidi e metalli pesanti nel terreno, possono influire notevolmente sulla capacità di ritenzione e lisciviazione. La flora, come le vulnerabili orchidee autoctone adattate a substrati poveri di nutrienti, può subire morie quando viene a contatto con sostanze chimiche di deflusso estranee che alterano il pH e la permeabilità del terreno. Un'analisi ha scoperto che una singola fuoriuscita di 500 galloni di acido nitrico ha portato all'estinzione regionale di una rara specie di muschio nell'anno successivo, quando l'effluente acido ha raggiunto gli habitat forestali vicini. Tali eventi di contaminazione innescano reazioni a catena attraverso la rete interconnessa della vita. Pertanto, protocolli rigorosi sono essenziali per evitare scarichi e deflussi pericolosi.

16.6 Analisi del Ciclo di Vita

Per comprendere l'impatto ambientale olistico dei sistemi di intelligenza artificiale è necessario un approccio completo che consideri l'intero ciclo di vita di queste tecnologie. Il "Life Cycle Analysis (LCA)" analisi del ciclo di vita si riferisce a un quadro metodologico utilizzato per quantificare gli impatti ambientali in tutte le fasi del ciclo di vita di un prodotto o sistema, dall'estrazione delle materie prime allo smaltimento a fine vita. L'applicazione dell'LCA ai sistemi di intelligenza artificiale può aiutare a identificare le aree prioritarie da prendere di mira per ridurre l'impatto ambientale complessivo.

16.6.1 Fasi del Ciclo di Vita di un Sistema di IA

Il ciclo di vita di un sistema di intelligenza artificiale può essere suddiviso in quattro fasi chiave:

- **Fase di Progettazione:** Include l'energia e le risorse utilizzate nella ricerca e nello sviluppo delle tecnologie di intelligenza artificiale. Comprende le risorse computazionali utilizzate per lo sviluppo e il test degli algoritmi che contribuiscono alle emissioni di carbonio.
- **Fase di Produzione:** Questa fase prevede la produzione di componenti hardware come schede grafiche, processori e altri dispositivi di elaborazione necessari per l'esecuzione degli algoritmi di intelligenza artificiale. La produzione di questi componenti spesso comporta un notevole consumo di energia per l'estrazione dei materiali, l'elaborazione e le emissioni di gas serra.
- **Fase di Utilizzo:** La fase successiva più dispendiosa in termini di energia riguarda l'uso operativo dei sistemi di intelligenza artificiale. Include l'elettricità consumata nei data center per l'addestramento e l'esecuzione delle reti neurali e l'alimentazione delle applicazioni per gli utenti finali. Questa è probabilmente una delle fasi con il più alto consumo di carbonio.

- **Fase di Smaltimento:** Questa fase finale riguarda gli aspetti di fine vita dei sistemi di intelligenza artificiale, tra cui il riciclaggio e lo smaltimento dei rifiuti elettronici generati da hardware obsoleto o non funzionante oltre la sua durata utile.

16.6.2 Impatto Ambientale in Ogni Fase

Progettazione e Produzione

L'impatto ambientale durante queste fasi iniziali di vita include emissioni derivanti dall'uso di energia e dall'esaurimento delle risorse derivanti dall'estrazione di materiali per la produzione di hardware. Al centro dell'hardware AI ci sono semiconduttori, principalmente silicio, utilizzati per realizzare i circuiti integrati nei processori e nei chip di memoria. Questa produzione di hardware si basa su metalli come il rame per i cablaggi, l'alluminio per gli involucri e varie plastiche e compositi per altri componenti. Utilizza anche terre rare e leghe specializzate, elementi come neodimio, terbio e ittrio, utilizzati in piccole ma vitali quantità. Ad esempio, la creazione di GPU si basa su rame e alluminio. Allo stesso tempo, i chip utilizzano terre rare, che è il processo di estrazione che può generare notevoli emissioni di carbonio e danni all'ecosistema.

Fase di Utilizzo

L'AI calcola la maggior parte delle emissioni nel ciclo di vita a causa del continuo elevato consumo di energia, in particolare per il training e l'esecuzione di modelli. Ciò include emissioni dirette e indirette derivanti dall'uso di elettricità e dalla generazione di energia di rete non rinnovabile. Gli studi stimano che l'addestramento di modelli complessi può avere un'impronta di carbonio paragonabile alle emissioni fino a cinque auto nel corso della loro vita.

Fase di Smaltimento

Gli impatti della fase di smaltimento includono l'inquinamento dell'aria e dell'acqua dovuto a materiali tossici nei dispositivi, le sfide associate al riciclaggio di componenti elettronici complessi e la contaminazione in caso di gestione impropria. I composti nocivi derivanti dalla combustione dei rifiuti elettronici vengono rilasciati nell'atmosfera. Allo stesso tempo, la perdita di piombo, mercurio e altri materiali dalle discariche comporta rischi di contaminazione del suolo e delle falde acquifere se non adeguatamente controllata. L'implementazione di un efficace riciclaggio dei componenti elettronici è fondamentale.

Esercizio 16.2: Monitoraggio delle Emissioni ML

In questo esercizio, esploreremo l'impatto ambientale dell'addestramento di modelli di machine learning. Utilizzeremo CodeCarbon per monitorare le emissioni, apprenderemo l'analisi del "Life Cycle Analysis (LCA)" per comprendere l'impronta di carbonio dell'IA ed esploreremo strategie per rendere lo sviluppo del modello ML più rispettoso dell'ambiente. Alla fine, si sarà in grado di monitorare le emissioni di carbonio dei modelli e iniziare a implementare pratiche più ecologiche nei progetti.

Open in Colab

16.7 Sfide nell'LCA

16.7.1 Mancanza di Coerenza e Standard

Una delle principali sfide che l'analisi del "life cycle analysis (LCA)" [ciclo di vita] deve affrontare per i sistemi di intelligenza artificiale è la necessità di standard e framework metodologici coerenti. A differenza di categorie di prodotti come i materiali da costruzione, che hanno sviluppato standard internazionali per LCA tramite ISO 14040, non esistono linee

guida stabilite per analizzare l'impatto ambientale di tecnologie informatiche complesse come l'intelligenza artificiale.
Questa assenza di uniformità significa che i ricercatori fanno ipotesi diverse e scelte metodologiche variabili. Ad esempio, uno studio del 2021 dell'Università del Massachusetts Amherst (Strubell, Ganesh, e McCallum 2019) ha analizzato le emissioni del ciclo di vita di diversi modelli di elaborazione del linguaggio naturale, ma ha preso in considerazione solo l'utilizzo delle risorse computazionali per il training e ha omesso gli impatti sulla produzione di hardware. Uno studio più completo del 2020 condotto dai ricercatori della Stanford University ha incluso stime delle emissioni derivanti dalla produzione di server, processori e altri componenti pertinenti, seguendo uno standard LCA allineato a ISO per l'hardware dei computer. Tuttavia, queste scelte divergenti nei confini del sistema e negli approcci contabili riducono la robustezza e impediscono confronti tra risultati simili.
Framework e protocolli standardizzati su misura per gli aspetti unici dei sistemi di intelligenza artificiale e rapidi cicli di aggiornamento fornirebbero maggiore coerenza. Ciò potrebbe consentire a ricercatori e sviluppatori di comprendere i punti critici ambientali, confrontare le opzioni tecnologiche e monitorare con precisione i progressi nelle iniziative di sostenibilità nel campo dell'intelligenza artificiale. Gruppi industriali e organismi di normazione internazionali come IEEE o ACM dovrebbero dare priorità all'affrontare questa lacuna metodologica.

16.7.2 Lacune nei Dati

Un'altra sfida fondamentale per una valutazione completa del ciclo di vita dei sistemi di intelligenza artificiale sono le lacune sostanziali nei dati, in particolare per quanto riguarda gli impatti sulla catena di fornitura a monte e i flussi di rifiuti elettronici a valle. La maggior parte degli studi esistenti si concentra strettamente sulle emissioni della fase di apprendimento o di utilizzo derivanti dalle richieste di potenza di calcolo, che tralasciano una parte significativa delle emissioni nel corso della vita (U. Gupta et al. 2022).
Ad esempio, esistono pochi dati pubblici delle aziende che quantificano l'uso di energia e le emissioni derivanti dalla produzione di componenti hardware specializzati che abilitano l'intelligenza artificiale, tra cui GPU di fascia alta, chip ASIC, unità a stato solido e altro. Spesso i ricercatori si affidano a fonti secondarie o medie generiche del settore per approssimare gli impatti sulla produzione. Analogamente, in media, c'è una trasparenza limitata sul destino a valle una volta che i sistemi di intelligenza artificiale vengono scartati dopo 4-5 anni di durata utile.
Mentre i livelli di generazione di rifiuti elettronici possono essere stimati, i dettagli sulle perdite di materiali pericolosi, sui tassi di riciclaggio e sui metodi di smaltimento per i componenti complessi sono estremamente incerti senza una migliore documentazione aziendale o requisiti di reporting normativi.
La necessità di dati dettagliati sul consumo di risorse computazionali per l'addestramento di diversi tipi di modelli rende difficili calcoli affidabili delle emissioni per parametro o per query anche per la fase di utilizzo. Esistono tentativi di creare inventari del ciclo di vita che stimino il fabbisogno energetico medio per le attività chiave dell'intelligenza artificiale (Henderson et al. 2020; Anthony, Kanding, e Selvan 2020), ma la variabilità tra configurazioni hardware, algoritmi e incertezza dei dati di input rimane estremamente elevata. Inoltre, i dati sull'intensità di carbonio in tempo reale, fondamentali per tracciare con precisione l'impronta di carbonio operativa, devono essere migliorati in molte posizioni geografiche, rendendo gli strumenti esistenti per le emissioni di carbonio operative mere approssimazioni basate sui valori di intensità di carbonio media annuale.
La sfida è che strumenti come CodeCarbon e ML CO_2 sono, nella migliore delle ipotesi, solo approcci ad hoc, nonostante le loro buone intenzioni. Colmare le lacune reali dei dati con

divulgazioni più rigorose sulla sostenibilità aziendale e una rendicontazione obbligatoria dell'impatto ambientale sarà fondamentale per comprendere e gestire gli impatti climatici complessivi dell'IA.

16.7.3 Rapido Ritmo di Evoluzione

L'evoluzione estremamente rapida dei sistemi di intelligenza artificiale pone ulteriori sfide nel mantenere aggiornate le valutazioni del ciclo di vita e nel tenere conto degli ultimi progressi hardware e software. Gli algoritmi di base, i chip specializzati, i framework e l'infrastruttura tecnica alla base dell'intelligenza artificiale hanno tutti fatto progressi eccezionalmente rapidi, con nuovi sviluppi che hanno rapidamente reso obsoleti i sistemi precedenti.
Ad esempio, nel deep learning, le nuove architetture di reti neurali che raggiungono prestazioni significativamente migliori su benchmark chiave o nuovi hardware ottimizzati come i chip TPU di Google possono cambiare completamente un modello "medio" in meno di un anno. Questi rapidi cambiamenti rendono rapidamente obsoleti gli studi LCA una tantum per il monitoraggio accurato delle emissioni derivanti dalla progettazione, esecuzione o smaltimento dell'intelligenza artificiale più recente.
Tuttavia, le risorse e l'accesso necessari per aggiornare continuamente gli LCA devono essere migliorati. Rifare frequentemente il lavoro, inventari del ciclo di vita ad alta intensità di dati e modelli di impatto per rimanere aggiornati con lo stato dell'arte dell'intelligenza artificiale è probabilmente irrealizzabile per molti ricercatori e organizzazioni. Tuttavia, analisi aggiornate potrebbero rilevare hotspot ambientali man mano che algoritmi e chip di silicio continuano a evolversi rapidamente.
Ciò presenta difficoltà nel bilanciare la precisione dinamica attraverso una valutazione continua con vincoli pragmatici. Alcuni ricercatori hanno proposto metriche proxy semplificate come il monitoraggio delle generazioni hardware nel tempo o l'utilizzo di benchmark rappresentativi come un set oscillante di paletti per confronti relativi, sebbene la granularità possa essere sacrificata. Nel complesso, la sfida del cambiamento rapido richiederà soluzioni metodologiche innovative per evitare di sottostimare gli oneri ambientali in evoluzione dell'IA.

16.7.4 Complessità della Catena di Fornitura

Infine, le complesse e spesso opache catene di fornitura associate alla produzione dell'ampia gamma di componenti hardware specializzati che abilitano l'intelligenza artificiale pongono sfide per la modellazione completa del ciclo di vita. Lo "stato-dell'arte" dellIA si basa su progressi all'avanguardia nell'elaborazione di chip, schede grafiche, archiviazione dati, apparecchiature di rete e altro ancora. Tuttavia, tracciare le emissioni e l'uso delle risorse attraverso le reti a livelli di fornitori globalizzati per tutti questi componenti è estremamente difficile.
Ad esempio, le unità di elaborazione grafica NVIDIA dominano gran parte dell'hardware di elaborazione dell'intelligenza artificiale, ma l'azienda si affida a diversi fornitori discreti in Asia e oltre per produrre GPU. Molte aziende a ogni livello di fornitore scelgono di mantenere privati i dati ambientali a livello di stabilimento, il che potrebbe abilitare completamente LCA robuste. Ottenere la trasparenza end-to-end su più livelli di fornitori in aree geografiche diverse con protocolli di divulgazione e normative variabili pone barriere nonostante sia fondamentale per la definizione completa dei confini. Ciò diventa ancora più complesso quando si tenta di modellare acceleratori hardware emergenti come le "tensor processing units (TPU)" [unità di elaborazione tensoriale], le cui reti di produzione devono ancora essere rese pubbliche.
Senza la volontà dei giganti della tecnologia di richiedere e consolidare la divulgazione dei dati sull'impatto ambientale da tutte le loro catene di fornitura di elettronica globali, rimarrà

una notevole incertezza sulla quantificazione dell'impronta del ciclo di vita completo dell'abilitazione hardware AI. Una maggiore visibilità della catena di fornitura abbinata a quadri di reporting sulla sostenibilità standardizzati che affrontino specificamente gli input complessi dell'AI promettono di arricchire gli LCA e dare priorità alle riduzioni dell'impatto ambientale.

16.8 Progettazione e Sviluppo Sostenibili

16.8.1 Principi di Sostenibilità

Man mano che l'impatto dell'IA sull'ambiente diventa sempre più evidente, l'attenzione alla progettazione e allo sviluppo sostenibili nell'IA sta acquisendo importanza. Ciò comporta l'incorporazione di principi di sostenibilità nella progettazione dell'IA, lo sviluppo di modelli a risparmio energetico e l'integrazione di queste considerazioni in tutta la pipeline di sviluppo dell'IA. C'è una crescente necessità di considerare le implicazioni di sostenibilità e sviluppare principi per guidare l'innovazione responsabile. Di seguito è riportato un set di principi fondamentali. I principi fluiscono dalle fondamenta concettuali all'esecuzione pratica ai fattori di supporto all'implementazione; i principi forniscono una prospettiva del ciclo completo sull'incorporamento della sostenibilità nella progettazione e nello sviluppo dell'IA.

Lifecycle Thinking: Incoraggiare i progettisti a considerare l'intero ciclo di vita dei sistemi di IA, dalla raccolta e preelaborazione dei dati allo sviluppo del modello, al training, all'implementazione e al monitoraggio. L'obiettivo è garantire che la sostenibilità sia presa in considerazione in ogni fase. Ciò include l'utilizzo di hardware a risparmio energetico, la priorità alle fonti di energia rinnovabili e la pianificazione del riutilizzo o del riciclaggio di modelli dismessi.

A Prova di Futuro: Progettare sistemi di intelligenza artificiale che anticipino esigenze e cambiamenti futuri può migliorare la sostenibilità. Ciò può comportare la creazione di modelli adattabili tramite apprendimento per trasferimento e architetture modulari. Include anche la capacità di pianificazione per aumenti previsti di scala operativa e volumi di dati.

Efficienza e Minimalismo: Questo principio si concentra sulla creazione di modelli di intelligenza artificiale che raggiungano i risultati desiderati con il minimo utilizzo di risorse possibile. Comporta la semplificazione di modelli e algoritmi per ridurre i requisiti computazionali. Tecniche specifiche includono la potatura di parametri ridondanti, la quantizzazione e la compressione di modelli e la progettazione di architetture di modelli efficienti, come quelle discusse nel capitolo Ottimizzazioni.

Integrazione del Lifecycle Assessment (LCA): [Valutazione del Ciclo di Vita] L'analisi degli impatti ambientali durante lo sviluppo e l'implementazione dei cicli di vita evidenzia le pratiche non sostenibili in anticipo. I team possono quindi apportare modifiche anziché scoprire i problemi in ritardo, quando sono più difficili da affrontare. L'integrazione di questa analisi nel flusso di progettazione standard evita di creare problemi ereditati di sostenibilità.

Allineamento degli Incentivi: Gli incentivi economici e politici dovrebbero promuovere e premiare lo sviluppo sostenibile dell'IA. Questi possono includere sovvenzioni governative, iniziative aziendali, standard di settore e mandati accademici per la sostenibilità. Gli incentivi allineati consentono alla sostenibilità di essere inglobata nella cultura dell'IA.

Metriche e Obiettivi di Sostenibilità: È importante stabilire metriche chiaramente definite che misurino fattori di sostenibilità come l'uso del carbonio e l'efficienza energetica. Stabilire obiettivi chiari per queste metriche fornisce linee guida concrete per i team per sviluppare

sistemi di IA responsabili. Il monitoraggio delle prestazioni sulle metriche nel tempo mostra i progressi verso gli obiettivi di sostenibilità prefissati.

Equità, Trasparenza e Responsabilità: I sistemi di IA sostenibili dovrebbero essere equi, trasparenti e responsabili. I modelli dovrebbero essere imparziali, con processi di sviluppo trasparenti e meccanismi per l'audit e la risoluzione dei problemi. Ciò crea fiducia nel pubblico e consente l'identificazione di pratiche non sostenibili.

Collaborazione Interdisciplinare: I ricercatori di intelligenza artificiale che collaborano con scienziati e ingegneri ambientali possono dare vita a sistemi innovativi ad alte prestazioni ma rispettosi dell'ambiente. L'unione di competenze provenienti da diversi campi fin dall'inizio dei progetti consente di incorporare il pensiero sostenibile nel processo di progettazione dell'intelligenza artificiale.

Istruzione e Consapevolezza: Workshop, programmi di formazione e programmi di studio che riguardano la sostenibilità dell'intelligenza artificiale accrescono la consapevolezza tra la prossima generazione di professionisti. Ciò fornisce agli studenti le conoscenze per sviluppare un'intelligenza artificiale che riduca al minimo gli impatti negativi sulla società e sull'ambiente. Inculcare questi valori fin dall'inizio plasma i professionisti e le culture aziendali di domani.

16.9 Infrastruttura di IA Green

Green AI rappresenta un approccio trasformativo all'IA che incorpora la sostenibilità ambientale come principio fondamentale nella progettazione e nel ciclo di vita del sistema di IA (R. Schwartz et al. 2020). Questo cambiamento è guidato dalla crescente consapevolezza dell'impatto ecologico e dell'impronta di carbonio significativa delle tecnologie di IA, in particolare il processo di elaborazione intensiva di modelli di ML complessi.
L'essenza di Green AI risiede nel suo impegno ad allineare il progresso dell'IA con gli obiettivi di sostenibilità in termini di efficienza energetica, utilizzo di energia rinnovabile e riduzione dei rifiuti. L'introduzione degli ideali di Green AI riflette la crescente responsabilità nel settore tecnologico verso la tutela ambientale e le pratiche tecnologiche etiche. Va oltre le ottimizzazioni tecniche verso una valutazione olistica del ciclo di vita su come i sistemi di IA influenzano le metriche di sostenibilità. Stabilire nuovi standard per un'IA ecologicamente consapevole apre la strada alla coesistenza armoniosa di progresso tecnologico e salute planetaria.

16.9.1 Sistemi di IA a Risparmio Energetico

L'efficienza energetica nei sistemi di intelligenza artificiale è un pilastro della Green AI, che mira a ridurre le richieste di energia tradizionalmente associate allo sviluppo e alle operazioni di intelligenza artificiale. Questo passaggio verso pratiche di intelligenza artificiale attente al risparmio energetico è fondamentale per affrontare le preoccupazioni ambientali sollevate dal campo in rapida espansione dell'intelligenza artificiale. Concentrandosi sull'efficienza energetica, i sistemi di intelligenza artificiale possono diventare più sostenibili, riducendo il loro impatto ambientale e aprendo la strada a un loro utilizzo più responsabile.
Come abbiamo discusso in precedenza, l'addestramento e il funzionamento dei modelli di intelligenza artificiale, in particolare quelli su larga scala, sono noti per il loro elevato consumo energetico, che deriva dall'architettura del modello ad alta intensità di calcolo e dall'affidamento a grandi quantità di dati di addestramento. Ad esempio, si stima che l'addestramento di un grande modello di rete neurale all'avanguardia possa avere un'impronta di carbonio di 284 tonnellate, equivalente alle emissioni di 5 auto nel corso della loro vita (Strubell, Ganesh, e McCallum 2019).

Per affrontare le enormi richieste di energia, ricercatori e sviluppatori stanno esplorando attivamente metodi per ottimizzare i sistemi di intelligenza artificiale per una migliore efficienza energetica mantenendo al contempo l'accuratezza e le prestazioni del modello. Ciò include tecniche come quelle che abbiamo discusso nei capitoli sulle ottimizzazioni del modello, sull'intelligenza artificiale efficiente e sull'accelerazione hardware:

- Distillazione della conoscenza per trasferire la conoscenza da grandi modelli di intelligenza artificiale a versioni in miniatura
- Approcci di quantizzazione e potatura che riducono le complessità computazionali e spaziali
- Numeri a bassa precisione: riduzione della precisione matematica senza influire sulla qualità del modello
- Hardware specializzato come TPU, chip neuromorfici ottimizzati esplicitamente per un'elaborazione efficiente dell'intelligenza artificiale

Un esempio è il lavoro di Intel su Q8BERT: quantizzazione del modello di linguaggio BERT con interi a 8 bit, che porta a una riduzione di 4 volte delle dimensioni del modello con una perdita di accuratezza minima (Zafrir et al. 2019). La spinta verso un'intelligenza artificiale efficiente dal punto di vista energetico non è solo uno sforzo tecnico: ha implicazioni tangibili nel mondo reale. Sistemi più performanti riducono i costi operativi e l'impatto ambientale dell'intelligenza artificiale, rendendola accessibile per un'ampia distribuzione su dispositivi mobili ed edge. Apre inoltre la strada alla democratizzazione dell'IA e mitiga i pregiudizi ingiusti che possono emergere da un accesso non uniforme alle risorse informatiche tra regioni e comunità. Perseguire un'IA efficiente dal punto di vista energetico è quindi fondamentale per creare un futuro equo e sostenibile con l'IA.

16.9.2 Infrastruttura di IA Sostenibile

L'infrastruttura AI sostenibile include i framework fisici e tecnologici che supportano i sistemi AI, concentrandosi sulla sostenibilità ambientale. Ciò implica la progettazione e la gestione dell'infrastruttura AI per ridurre al minimo l'impatto ecologico, conservare le risorse e ridurre le emissioni di carbonio. L'obiettivo è creare un ecosistema sostenibile per l'AI che si allinei con obiettivi ambientali più ampi.

I data center green sono fondamentali per l'infrastruttura AI sostenibile, ottimizzati per l'efficienza energetica e spesso alimentati da fonti di energia rinnovabili. Questi data center impiegano tecnologie di raffreddamento avanzate (Ebrahimi, Jones, e Fleischer 2014), design di server a risparmio energetico (Uddin e Rahman 2012) e sistemi di gestione intelligenti (Buyya, Beloglazov, e Abawajy 2010) per ridurre il consumo di energia. Il passaggio a un'infrastruttura informatica verde implica anche l'adozione di hardware a risparmio energetico, come processori ottimizzati per l'AI che offrono prestazioni elevate con requisiti energetici inferiori, di cui abbiamo discusso nel capitolo Accelerazione IA. Questi sforzi riducono collettivamente l'impronta di carbonio delle operazioni di intelligenza artificiale su larga scala.

L'integrazione di fonti di energia rinnovabili, come energia solare, eolica e idroelettrica, nell'infrastruttura di intelligenza artificiale è importante per la sostenibilità ambientale (Chua 1971). Molte aziende tecnologiche e istituti di ricerca stanno investendo in progetti di energia rinnovabile per alimentare i loro data center. Ciò non solo aiuta a rendere le operazioni di intelligenza artificiale carbon neutral, ma promuove anche un'adozione più ampia di energia pulita. L'utilizzo di fonti di energia rinnovabili mostra chiaramente l'impegno per la responsabilità ambientale nel settore dell'intelligenza artificiale.

La sostenibilità si estende anche ai materiali e all'hardware utilizzati nella creazione di sistemi di intelligenza artificiale. Ciò implica la scelta di materiali ecocompatibili, l'adozione

di pratiche di riciclaggio e la garanzia di uno smaltimento responsabile dei rifiuti elettronici. Sono in corso sforzi per sviluppare componenti hardware più sostenibili, tra cui chip a risparmio energetico progettati per attività specifiche del dominio (come gli acceleratori di IA) e materiali ecocompatibili nella produzione di dispositivi (Cenci et al. 2021; Irimia-Vladu 2014). Anche il ciclo di vita di questi componenti è un punto focale, con iniziative volte a estendere la durata di vita dell'hardware e a promuovere il riciclaggio e il riutilizzo.
Sebbene si stiano facendo progressi nell'infrastruttura di IA sostenibile, permangono delle sfide, come gli elevati costi della tecnologia verde e la necessità di standard globali nelle pratiche sostenibili. Le direzioni future includono un'adozione più diffusa di energia verde, ulteriori innovazioni nell'hardware a risparmio energetico e la collaborazione internazionale su politiche di IA sostenibile. Perseguire un'infrastruttura di IA sostenibile non è solo uno sforzo tecnico, ma un approccio olistico che comprende aspetti ambientali, economici e sociali, assicurando che l'IA avanzi in armonia con la salute del nostro pianeta.

16.9.3 Framework e Strumenti

L'accesso ai framework e agli strumenti giusti è essenziale per implementare in modo efficace le pratiche di intelligenza artificiale verde. Queste risorse sono progettate per aiutare sviluppatori e ricercatori a creare sistemi di IA più efficienti dal punto di vista energetico e rispettosi dell'ambiente. Vanno da librerie software ottimizzate per un basso consumo energetico a piattaforme che facilitano lo sviluppo di applicazioni di IA sostenibili.
Diverse librerie software e ambienti di sviluppo sono specificamente pensati per l'intelligenza artificiale verde. Questi strumenti spesso includono funzionalità per ottimizzare i modelli di IA per ridurre il loro carico computazionale e, di conseguenza, il loro consumo energetico. Ad esempio, le librerie in PyTorch e TensorFlow che supportano la potatura del modello, la quantizzazione e le architetture di reti neurali efficienti consentono agli sviluppatori di creare sistemi di intelligenza artificiale che richiedono meno potenza di elaborazione ed energia. Inoltre, comunità open source come la Green Software Foundation stanno creando una metrica centralizzata dell'intensità di carbonio e sviluppando software per un'informatica attenta alle emissioni di carbonio.
Gli strumenti di monitoraggio dell'energia sono fondamentali per l'intelligenza artificiale verde, poiché consentono agli sviluppatori di misurare e analizzare il consumo energetico dei loro sistemi. Figura 16.8 è uno screenshot di una dashboard del consumo energetico fornita dalla piattaforma di servizi cloud di Microsoft. Fornendo informazioni dettagliate su dove e come viene utilizzata l'energia, questi strumenti consentono agli sviluppatori di prendere decisioni informate sull'ottimizzazione dei loro modelli per una migliore efficienza energetica. Ciò può comportare modifiche nella progettazione dell'algoritmo, nella selezione dell'hardware, nella selezione del software di cloud computing o nei parametri operativi.

Figura 16.8: Dashboard del consumo energetico di Microsoft Azure. Fonte: Will Buchanan.

Con la crescente integrazione di fonti di energia rinnovabile nelle operazioni di IA, i framework che facilitano questo processo stanno diventando sempre più importanti. Questi framework aiutano a gestire l'approvvigionamento energetico da fonti rinnovabili come l'energia solare o

eolica, assicurando che i sistemi di IA possano funzionare in modo efficiente con input energetici fluttuanti.
Oltre all'efficienza energetica, gli strumenti di valutazione della sostenibilità aiutano a valutare l'impatto ambientale più ampio dei sistemi di IA. Questi strumenti possono analizzare fattori come l'impronta di carbonio delle operazioni di IA, l'impatto del ciclo di vita dei componenti hardware (U. Gupta et al. 2022) e la sostenibilità complessiva dei progetti di IA (Prakash, Callahan, et al. 2023).
La disponibilità e lo sviluppo continuo di framework e strumenti di IA Green sono fondamentali per promuovere pratiche di sostenibili. Fornendo le risorse necessarie a sviluppatori e ricercatori, questi strumenti facilitano la creazione di sistemi più rispettosi dell'ambiente e incoraggiano un più ampio cambiamento verso la sostenibilità nella comunità tecnologica. Man mano che l'IA Green continua a evolversi, questi framework e strumenti svolgeranno un ruolo fondamentale nel dare forma a un futuro più sostenibile per l'IA. As Green AI continues to evolve, these frameworks and tools will play a vital role in shaping a more sustainable future for AI.

16.9.4 Benchmark e Classifiche

Benchmark e classifiche sono importanti per guidare i progressi nell'IA Green, poiché forniscono modi standardizzati per misurare e confrontare diversi metodi. Benchmark ben progettati che catturano metriche rilevanti su efficienza energetica, emissioni di carbonio e altri fattori di sostenibilità consentono alla comunità di monitorare i progressi in modo equo e significativo.
Esistono ampi benchmark per tracciare le prestazioni del modello di IA, come quelli discussi nel capitolo Benchmarking. Tuttavia, esiste una chiara e urgente necessità di ulteriori benchmark standardizzati incentrati su parametri di sostenibilità come efficienza energetica, emissioni di carbonio e impatto ecologico complessivo. La comprensione dei costi ambientali dell'IA deve attualmente essere migliorata da una mancanza di trasparenza e misura standardizzata attorno a questi fattori.
Sforzi emergenti come ML.ENERGY Leaderboard, che fornisce risultati di benchmarking delle prestazioni e del consumo energetico per la generazione di testo di modelli linguistici di grandi dimensioni (LLM), aiutano a migliorare la comprensione del costo energetico dell'implementazione GenAI.
Come con qualsiasi benchmark, quelli di IA Green devono rappresentare scenari di utilizzo e carichi di lavoro realistici. I benchmark che si concentrano strettamente su metriche facilmente manipolabili possono portare a guadagni a breve termine, ma non riescono a riflettere gli ambienti di produzione effettivi in cui sono necessarie misure di efficienza e sostenibilità più olistiche. La comunità dovrebbe continuare ad espandere i benchmark per coprire diversi casi d'uso.
Un'adozione più ampia di suite di benchmark comuni da parte degli operatori del settore accelererà l'innovazione nell'IA Green consentendo un confronto più semplice delle tecniche tra le organizzazioni. I benchmark condivisi abbassano la barriera per dimostrare i vantaggi di sostenibilità di nuovi strumenti e best practice. uttavia, quando si progettano benchmark per l'intero settore, è necessario prestare attenzione a questioni come proprietà intellettuale, privacy e sensibilità commerciale. Le iniziative per sviluppare set di dati di riferimento aperti per la valutazione dell'IA Green possono aiutare a promuovere una partecipazione più ampia.
Man mano che i metodi e l'infrastruttura per l'IA Green continuano a maturare, la comunità deve rivedere la progettazione dei benchmark per garantire che le suite esistenti catturino bene nuove tecniche e scenari. Monitorare il panorama in evoluzione attraverso aggiornamenti e revisioni regolari dei benchmark sarà importante per mantenere confronti rappresentativi nel tempo. Gli sforzi della comunità per la cura dei benchmark possono

consentire suite di benchmark sostenibili che resistano alla prova del tempo. Suite di benchmark complete di proprietà di comunità di ricerca o terze parti neutrali come MLCommons possono incoraggiare una più ampia partecipazione e standardizzazione.

16.10 Caso di Studio: 4M di Google

Negli ultimi dieci anni, l'intelligenza artificiale è passata rapidamente dalla ricerca accademica ai sistemi di produzione su larga scala che alimentano numerosi prodotti e servizi Google. Poiché i modelli e i carichi di lavoro dell'IA sono cresciuti esponenzialmente in termini di dimensioni e richieste di elaborazione, sono emerse preoccupazioni circa il loro consumo energetico e l'impatto ambientale. Alcuni ricercatori hanno previsto una crescita incontrollata dell'appetito energetico del ML che potrebbe superare le efficienze ottenute da algoritmi e hardware migliorati (Thompson et al. 2021).
Tuttavia, i dati di produzione di Google rivelano una storia diversa: l'IA rappresenta un costante 10-15% del consumo energetico totale dell'azienda dal 2019 al 2021. Questo caso di studio analizza come Google ha applicato un approccio sistematico sfruttando quattro best practice, quelle che definiscono le "4 M": "Model efficiency", "Machine optimization", "Mechanization through cloud computing" e "Mapping to green locations" [efficienza del modello, ottimizzazione delle macchine, meccanizzazione tramite cloud computing e mappatura di luoghi Green], per piegare la curva delle emissioni dai carichi di lavoro dell'IA.
La portata dell'utilizzo dell'IA da parte di Google lo rende un caso di studio ideale. Solo nel 2021, l'azienda ha addestrato modelli come il GLam da 1,2 trilioni di parametri. Analizzare come l'applicazione dell'IA è stata abbinata a rapidi guadagni di efficienza in questo ambiente ci aiuta a fornire un modello logico che il più ampio campo dell'IA seguirà.
Pubblicando in modo trasparente statistiche dettagliate sull'uso dell'energia, adottando tassi di acquisto di cloud senza emissioni di carbonio e fonti rinnovabili e altro ancora, insieme alle sue innovazioni tecniche, Google ha consentito ai ricercatori esterni di misurare i progressi in modo accurato. Il loro studio nell'ACM CACM (D. Patterson et al. 2022) evidenzia come l'approccio multiforme dell'azienda dimostri che le previsioni di consumo energetico dell'IA incontrollabili possono essere superate concentrando gli sforzi ingegneristici su modelli di sviluppo sostenibile. Il ritmo dei miglioramenti suggerisce anche che i guadagni di efficienza dell'ML sono appena iniziati.

16.10.1 Le 4M Best Practice di Google

Per ridurre le emissioni derivanti dai carichi di lavoro IA in rapida espansione, gli ingegneri di Google hanno sistematicamente identificato quattro aree di best practice, denominate "4 M", in cui le ottimizzazioni potrebbero sommarsi per ridurre l'impatto ambientale del ML:

- **Modello:** La selezione di architetture di modelli di intelligenza artificiale efficienti può ridurre i calcoli di 5-10 volte senza alcuna perdita di qualità. Google ha svolto ricerche approfondite sullo sviluppo di modelli "sparsi" e sulla ricerca di architetture neurali per creare modelli più efficienti come Evolved Transformer e Primer.
- **Macchina:** L'utilizzo di hardware ottimizzato per l'IA rispetto ai sistemi generici migliora le prestazioni per watt di 2-5 volte. Le Tensor Processing Unit (TPU) di Google hanno portato a un'efficienza di carbonio 5-13 volte migliore rispetto alle GPU non ottimizzate per il ML.
- **Meccanizzazione:** Sfruttando i sistemi di cloud computing progettati per un utilizzo elevato rispetto ai tradizionali data center on-premise, i costi energetici si riducono di 1,4-2 volte. Google cita l'efficacia dell'utilizzo energetico del suo data center come superiore alle medie del settore.
- **Mappa:** La scelta di ubicazioni per data center dotate di elettricità a basse emissioni di carbonio riduce le emissioni lorde di altre 5-10 volte. Google fornisce mappe in

tempo reale che evidenziano la percentuale di energia rinnovabile utilizzata dalle sue strutture.

Insieme, queste pratiche hanno creato drastici guadagni di efficienza composti. Ad esempio, l'ottimizzazione del modello Transformer AI su TPU in una sede di data center sostenibile ha ridotto il consumo di energia dell'83x. Ha ridotto le emissioni di CO_2 di un fattore di 747.

16.10.2 Risultati Significativi

Nonostante la crescita esponenziale nell'adozione dell'IA nei prodotti e nei servizi, gli sforzi di Google per migliorare l'efficienza del carbonio del ML hanno prodotto guadagni misurabili, contribuendo a limitare l'appetito energetico complessivo. Un punto dati chiave che evidenzia questo progresso è che i carichi di lavoro dell'intelligenza artificiale sono rimasti stabili al 10%-15% del consumo energetico totale dell'azienda dal 2019 al 2021. Man mano che l'IA è diventata parte integrante di più offerte Google, i cicli di elaborazione complessivi dedicati ad essa sono cresciuti in modo sostanziale. Tuttavia, l'efficienza negli algoritmi, nell'hardware specializzato, nella progettazione dei data center e nella geografia flessibile ha consentito alla sostenibilità di tenere il passo, con l'IA che rappresenta solo una frazione dell'elettricità totale del data center in anni di espansione.

Altri casi di studio sottolineano come un focus ingegneristico sui pattern di sviluppo dell'intelligenza artificiale sostenibile abbia consentito rapidi miglioramenti della qualità di pari passo con i guadagni ambientali. Ad esempio, il modello di elaborazione del linguaggio naturale GPT-3 è stato considerato all'avanguardia a metà del 2020. Tuttavia, il suo successore GLaM ha migliorato la precisione riducendo al contempo le esigenze di elaborazione del training e utilizzando energia più pulita nei data center, riducendo le emissioni di CO_2 di un fattore 14 in soli 18 mesi di evoluzione del modello.

Analogamente, Google ha scoperto che le precedenti speculazioni pubblicate non hanno colto nel segno sull'appetito energetico del ML per fattori da 100 a 100.000X a causa della mancanza di metriche del mondo reale. Tracciando in modo trasparente l'impatto dell'ottimizzazione, Google sperava di motivare l'efficienza evitando al contempo estrapolazioni sovrastimate sul pedaggio ambientale del ML.

Questi casi di studio basati sui dati mostrano come aziende come Google stiano indirizzando i progressi dell'IA verso traiettorie sostenibili e migliorando l'efficienza per superare la crescita dell'adozione. Con ulteriori sforzi in termini di analisi del ciclo di vita, ottimizzazione dell'inferenza ed espansione delle energie rinnovabili, le aziende possono puntare ad accelerare i progressi, dimostrando che il potenziale pulito del ML è stato appena sbloccato dagli attuali guadagni.

16.10.3 Ulteriori Miglioramenti

Sebbene Google abbia compiuto progressi misurabili nel limitare l'impatto ambientale delle sue operazioni di intelligenza artificiale, l'azienda riconosce che ulteriori guadagni in termini di efficienza saranno essenziali per un'innovazione responsabile, data la continua espansione della tecnologia.

Un'area di attenzione è mostrare come i progressi siano spesso erroneamente considerati come un aumento dell'insostenibilità informatica, come la ricerca di architettura neurale (NAS) per trovare modelli ottimizzati, che stimolano risparmi a valle, superando i costi iniziali. Nonostante spenda più energia nella scoperta di modelli piuttosto che nell'ingegneria manuale, la NAS riduce le emissioni nel corso del ciclo di vita producendo progetti efficienti richiamabili su innumerevoli applicazioni.

Inoltre, l'analisi rivela che concentrare gli sforzi di sostenibilità sull'ottimizzazione lato server e data center ha senso, dato il consumo energetico dominante rispetto ai dispositivi consumer. Sebbene Google riduca gli impatti dell'inferenza su processori come i telefoni

cellulari, la priorità è il miglioramento dei cicli di training e dell'approvvigionamento di energie rinnovabili per data center per ottenere il massimo effetto.
A tal fine, i progressi di Google nel mettere in comune strutture cloud progettate in modo inefficiente evidenziano il valore della scala e della centralizzazione. Con l'allontanamento dei carichi di lavoro dai server locali inefficienti, la priorità data dai giganti di Internet alle energie rinnovabili (con Google e Meta che hanno raggiunto il 100% di energie rinnovabili rispettivamente dal 2017 e dal 2020) sblocca tagli alle emissioni complessive.
Insieme, questi sforzi sottolineano che, sebbene non sia possibile adagiarsi sugli allori, l'approccio multiforme di Google dimostra che i miglioramenti dell'efficienza dell'IA stanno solo accelerando. Le iniziative intersettoriali relative alla valutazione del ciclo di vita, ai pattern di sviluppo attenti alle emissioni di carbonio, alla trasparenza e all'abbinamento della crescente domanda di IA con la fornitura di energia elettrica pulita aprono la strada a un'ulteriore flessione della curva man mano che l'adozione aumenta. I risultati dell'azienda spingono il settore più ampio a replicare queste attività di sostenibilità integrate.

16.11 IA Embedded - Internet of Trash

Sebbene molta attenzione sia stata rivolta a rendere più sostenibili gli immensi data center che alimentano l'IA, una preoccupazione altrettanto urgente è lo spostamento delle capacità dell'IA in dispositivi edge e endpoint intelligenti. L'IA edge/embedded consente una reattività quasi in tempo reale senza dipendenze dalla connettività. Riduce inoltre le esigenze di larghezza di banda di trasmissione. Tuttavia, l'aumento di dispositivi minuscoli comporta altri rischi.
I minuscoli computer, microcontrollori e ASIC personalizzati che alimentano l'intelligenza edge affrontano limitazioni di dimensioni, costi e potenza che escludono le GPU di fascia alta utilizzate nei data center. Invece, richiedono algoritmi ottimizzati e circuiti estremamente compatti ed efficienti dal punto di vista energetico per funzionare senza problemi. Tuttavia, l'ingegneria per questi fattori di forma microscopici apre rischi in termini di obsolescenza programmata, smaltibilità e spreco. Figura 16.9 mostra che si prevede che il numero di dispositivi IoT raggiungerà i 30 miliardi di dispositivi connessi entro il 2030.

Figura 16.9: Numero di dispositivi connessi all'Internet of Things (IoT) in tutto il mondo dal 2019 al 2023. Fonte: Statista.

La gestione del fine vita dei gadget connessi a Internet dotati di sensori e intelligenza artificiale rimane un problema spesso trascurato durante la progettazione. Tuttavia, questi prodotti permeano beni di consumo, veicoli, infrastrutture pubbliche, apparecchiature industriali e altro ancora.

16.11.1 Rifiuti Elettronici

I rifiuti elettronici, o "e-waste", si riferiscono ad apparecchiature elettriche e componenti scartati che entrano nel flusso dei rifiuti. Ciò include dispositivi che devono essere collegati, hanno una batteria o circuiti elettrici. Con la crescente adozione di dispositivi intelligenti e sensori connessi a Internet, i volumi di e-waste aumentano rapidamente ogni anno. Questi gadget in proliferazione contengono metalli pesanti tossici come piombo, mercurio e cadmio che diventano pericoli per l'ambiente e la salute se smaltiti in modo improprio.

La quantità di rifiuti elettronici prodotti sta crescendo a un ritmo allarmante. Oggi, ne produciamo già 50 milioni di tonnellate all'anno. Entro il 2030, si prevede che tale cifra salirà a ben 75 milioni di tonnellate, poiché il consumo di elettronica di consumo continua ad accelerare. La produzione globale di e-waste raggiungerà i 120 milioni di tonnellate all'anno entro il 2050 (Un e Forum 2019). La produzione in forte crescita e i brevi cicli di vita dei nostri gadget alimentano questa crisi, dagli smartphone e tablet ai dispositivi connessi a Internet e agli elettrodomestici.

I paesi in via di sviluppo sono i più colpiti, in quanto necessitano di più infrastrutture per elaborare in modo sicuro i dispositivi elettronici obsoleti. Nel 2019, i tassi di riciclaggio formali dei rifiuti elettronici nei paesi più poveri variavano dal 13% al 23%. Il resto finisce per essere scaricato illegalmente, bruciato o smantellato in modo grossolano, rilasciando materiali tossici nell'ambiente e danneggiando i lavoratori e le comunità locali. Chiaramente, c'è ancora molto da fare per costruire una capacità globale per una gestione etica e sostenibile dei rifiuti elettronici, altrimenti rischiamo danni irreversibili.

Il pericolo è che la manipolazione grossolana dei dispositivi elettronici per spogliarli delle parti di valore esponga i lavoratori e le comunità emarginati a nocive plastiche/metalli bruciati. L'avvelenamento da piombo presenta rischi particolarmente elevati per lo sviluppo infantile se ingerito o inalato. Nel complesso, solo circa il 20% dei rifiuti elettronici prodotti è stato raccolto utilizzando metodi ecologicamente corretti, secondo le stime delle Nazioni Unite (Un e Forum 2019). Quindi sono urgentemente necessarie soluzioni per una gestione responsabile del ciclo di vita per contenere lo smaltimento non sicuro, dato che il volume aumenta vertiginosamente.

16.11.2 Elettronica Monouso

I costi in rapida diminuzione dei microcontrollori, delle piccole batterie ricaricabili e dell'hardware compatto di comunicazione hanno consentito l'integrazione di sistemi di sensori intelligenti nei beni di consumo di uso quotidiano. Questi dispositivi Internet-of-Things (IoT) monitorano le condizioni del prodotto, le interazioni degli utenti e i fattori ambientali per consentire reattività in tempo reale, personalizzazione e decisioni aziendali basate sui dati nel mercato connesso in evoluzione.

Tuttavia, questi dispositivi elettronici integrati affrontano poca supervisione o pianificazione per gestire in modo sostenibile il loro eventuale smaltimento una volta che i prodotti spesso rivestiti in plastica vengono scartati dopo breve tempo. I sensori IoT ora risiedono comunemente in articoli monouso come bottiglie d'acqua, imballaggi per alimenti, flaconi di farmaci e contenitori per cosmetici che finiscono prevalentemente nei flussi di rifiuti delle discariche dopo poche settimane o mesi di utilizzo da parte dei consumatori.

Il problema si accelera poiché sempre più produttori si affrettano a integrare chip mobili, fonti di alimentazione, moduli Bluetooth e altri moderni circuiti integrati in silicio, che

costano meno di 1 dollaro USA, in vari prodotti senza protocolli per il riciclaggio, la sostituzione delle batterie o la riutilizzabilità dei componenti. Nonostante le loro piccole dimensioni individuali, i volumi di questi dispositivi e il peso dei rifiuti nel corso della loro vita incombono. A differenza della regolamentazione di dispositivi elettronici più grandi, esistono pochi vincoli normativi sui requisiti dei materiali o sulla tossicità di piccoli gadget monouso. Pur offrendo praticità durante il lavoro, la combinazione insostenibile di difficile recupero e limitati meccanismi di guasto sicuri fa sì che i dispositivi connessi monouso contribuiscano a quote sproporzionate di futuri volumi di rifiuti elettronici che necessitano di urgente attenzione.

16.11.3 Obsolescenza Programmata

L'obsolescenza programmata si riferisce alla strategia di progettazione intenzionale di produzione di prodotti con durate di vita artificialmente limitate che diventano rapidamente non funzionali o obsoleti. Ciò stimola cicli di acquisto di sostituzione più rapidi poiché i consumatori scoprono che i dispositivi non soddisfano più le loro esigenze nel giro di pochi anni. Tuttavia, l'elettronica progettata per l'obsolescenza prematura contribuisce a volumi di rifiuti elettronici insostenibili.

Ad esempio, incollare batterie e componenti di smartphone insieme ostacola la riparabilità rispetto ad assemblaggi modulari e accessibili. L'implementazione di aggiornamenti software che rallentano deliberatamente le prestazioni del sistema crea la percezione che valga la pena aggiornare i dispositivi prodotti solo diversi anni prima.

Allo stesso modo, le introduzioni alla moda di nuove generazioni di prodotti con aggiunte di funzionalità minori ma esclusive fanno sembrare rapidamente datate le versioni precedenti. Queste tattiche costringono ad acquistare nuovi gadget (ad esempio, iPhone) molto prima della fine della loro operatività. Se moltiplicati per categorie di elettronica in rapida evoluzione, miliardi di articoli appena indossati vengono scartati ogni anno.

L'obsolescenza programmata intensifica quindi l'utilizzo delle risorse e la creazione di rifiuti nella produzione di prodotti senza alcuna intenzione di lunga durata. Ciò contraddice i principi di sostenibilità in materia di durata, riutilizzo e conservazione dei materiali. Mentre stimola vendite e guadagni continui per i produttori nel breve termine, la strategia esternalizza costi ambientali e tossine su comunità prive di un'adeguata infrastruttura di elaborazione dei rifiuti elettronici.

Le politiche e l'azione dei consumatori sono fondamentali per contrastare i design dei gadget che sono inutilmente monouso per default. Le aziende dovrebbero anche investire in programmi di gestione dei prodotti che supportino il riutilizzo e il recupero responsabili.

Consideriamo l'esempio del mondo reale. Apple è stata attenzionata nel corso degli anni per aver presumibilmente coinvolto nell'obsolescenza programmata per incoraggiare i clienti ad acquistare nuovi modelli di iPhone. L'azienda avrebbe progettato i suoi telefoni in modo che le prestazioni si degradino nel tempo o che le funzionalità esistenti diventino incompatibili con i nuovi sistemi operativi, il che, secondo i critici, è finalizzato a stimolare cicli di aggiornamento più rapidi. Nel 2020, Apple ha pagato una multa di 25 milioni di euro per risolvere un caso in Francia in cui le autorità di regolamentazione hanno ritenuto l'azienda colpevole di aver rallentato intenzionalmente i vecchi iPhone senza informare chiaramente i clienti tramite aggiornamenti di iOS.

Non essendo trasparente sulle modifiche alla gestione dell'alimentazione che hanno ridotto le prestazioni del dispositivo, Apple ha partecipato ad attività ingannevoli che hanno ridotto la durata del prodotto per aumentare le vendite. L'azienda ha affermato che è stato fatto per "smussare" i picchi che potrebbero causare improvvisamente lo spegnimento delle vecchie batterie. Tuttavia, questo esempio evidenzia i rischi legali legati all'impiego dell'obsolescenza programmata e alla mancata comunicazione corretta di quando le modifiche alle funzionalità

influiscono sull'usabilità del dispositivo nel tempo: persino marchi leader come Apple possono avere problemi se percepiti come coloro che accorciano intenzionalmente i cicli di vita del prodotto.

16.12 Considerazioni Normative e Politiche

16.12.1 Mandati di Misura e Rendicontazione

Un meccanismo politico sempre più rilevante per i sistemi di IA è rappresentato dai requisiti di misurazione e rendicontazione relativi al consumo energetico e alle emissioni di carbonio. Misurazioni obbligatorie, audit, divulgazioni e metodologie più rigorose allineate alle metriche di sostenibilità possono aiutare a colmare le lacune informative che ostacolano le ottimizzazioni dell'efficienza.

Allo stesso tempo, le politiche nazionali o regionali richiedono alle aziende di una certa dimensione di utilizzare l'IA nei loro prodotti o sistemi back-end per segnalare il consumo energetico o le emissioni associate ai principali carichi di lavoro di IA. Organizzazioni come la Partnership on AI, IEEE e NIST potrebbero contribuire a definire metodologie standardizzate. Proposte più complesse implicano la definizione di modalità coerenti per misurare la complessità computazionale, il PUE del data center, l'intensità di carbonio dell'approvvigionamento energetico e le efficienze ottenute tramite hardware specifico per l'IA.

Anche gli obblighi di rendicontazione per gli utenti del settore pubblico che acquistano servizi di IA, ad esempio tramite una proposta di legge in Europa, potrebbero aumentare la trasparenza. Tuttavia, gli enti regolatori devono bilanciare l'ulteriore onere di misurazione che tali mandati impongono alle organizzazioni con le continue riduzioni di carbonio derivanti dall'incorporazione di pattern di sviluppo consapevoli della sostenibilità.

Per essere più costruttivi, qualsiasi politica di misurazione e rendicontazione dovrebbe concentrarsi sull'abilitazione di un continuo perfezionamento piuttosto che su restrizioni o limiti semplicistici. Man mano che i progressi dell'IA si sviluppano rapidamente, agili barriere di sicurezza di governance che incorporano considerazioni sulla sostenibilità in normali metriche di valutazione possono motivare un cambiamento positivo. Tuttavia, una prescrizione eccessiva rischia di limitare l'innovazione se i requisiti diventano obsoleti. La politica di efficienza dell'IA accelera i progressi in tutto il settore combinando flessibilità con appropriate barriere di sicurezza di trasparenza.

16.12.2 Meccanismi di Restrizione

Oltre agli obblighi di segnalazione, i politici dispongono di diversi meccanismi di restrizione che potrebbero modellare direttamente il modo in cui i sistemi di IA vengono sviluppati e implementati per ridurre le emissioni:

Limiti sulle Emissioni delle Elaborazioni: La proposta di legge sull'intelligenza artificiale della Commissione europea adotta un approccio orizzontale che potrebbe consentire di stabilire limiti per l'intera economia sul volume di potenza di elaborazione disponibile per l'addestramento dei modelli di intelligenza artificiale. Come i sistemi di scambio delle emissioni, i limiti mirano a disincentivare indirettamente l'elaborazione estensiva rispetto alla sostenibilità. Tuttavia, la qualità del modello potrebbe essere migliorata per fornire più percorsi per l'acquisizione di capacità aggiuntiva.

Condizionamento dell'Accesso alle Risorse Pubbliche: Alcuni esperti hanno proposto incentivi come consentire l'accesso solo a set di dati pubblici o potenza di elaborazione per lo sviluppo di modelli fondamentalmente efficienti piuttosto che architetture stravaganti. Ad esempio, il consorzio di benchmarking MLCommons fondato da importanti aziende tecnologiche potrebbe integrare formalmente l'efficienza nelle sue metriche di classifica standardizzate, tuttavia, l'accesso condizionato rischia di limitare l'innovazione.

Meccanismi Finanziari: Analogamente alle tasse sul carbonio sulle industrie inquinanti, le tariffe applicate per unità di consumo di elaborazione correlato all'IA potrebbero scoraggiare un inutile ridimensionamento del modello, finanziando al contempo innovazioni di efficienza. I crediti d'imposta potrebbero in alternativa premiare le organizzazioni pioniere di tecniche di IA più accurate ma compatte. Tuttavia, gli strumenti finanziari richiedono un'attenta calibrazione tra generazione di entrate ed equità e non penalizzare eccessivamente gli usi produttivi dell'IA.
Divieti Tecnologici: Se la misurazione fissasse costantemente le emissioni estreme su applicazioni specifiche dell'IA senza percorsi di bonifica, i divieti assoluti rappresentano uno strumento di ultima istanza per i decisori politici. Tuttavia, dato il duplice uso dell'IA, definire implementazioni dannose e benefiche risulta complesso, rendendo necessaria una valutazione di impatto olistica prima di concludere che non esiste alcun valore redentivo. Vietare tecnologie promettenti rischia di avere conseguenze indesiderate e richiede cautela.

16.12.3 Incentivi Governativi

È una pratica comune per i governi fornire incentivi fiscali o di altro tipo a consumatori o aziende quando contribuiscono a pratiche tecnologiche più sostenibili. Tali incentivi esistono già negli Stati Uniti per l'adozione di pannelli solari o edifici a risparmio energetico. Per quanto ne sappiamo, non esistono ancora incentivi fiscali per pratiche di sviluppo specifiche per l'IA.
Un altro potenziale programma di incentivi che sta iniziando a essere esplorato è l'utilizzo di sovvenzioni governative per finanziare progetti di IA Green. Ad esempio, in Spagna, sono stati stanziati 300 milioni di euro per finanziare specificamente progetti di IA e sostenibilità. Gli incentivi governativi sono una strada promettente per incoraggiare pratiche di comportamento aziendale e dei consumatori sostenibili, ma è necessaria un'attenta riflessione per determinare come tali incentivi si adatteranno alle richieste del mercato (Cohen, Lobel, e Perakis 2016).

16.12.4 Autoregolamentazione

Complementari alle potenziali azioni governative, i meccanismi di autogoverno volontario consentono alla comunità dell'IA di perseguire obiettivi di sostenibilità senza interventi dall'alto:
Impegni per le Energie Rinnovabili: Grandi professionisti dell'IA come Google, Microsoft, Amazon e Meta si sono impegnati ad acquistare abbastanza elettricità rinnovabile per soddisfare il 100% delle loro richieste energetiche. Questi impegni sbloccano tagli alle emissioni composti man mano che aumenta la potenza di calcolo. La formalizzazione di tali programmi incentiva le regioni dei data center verdi. Tuttavia, ci sono critiche sul fatto che questi impegni siano sufficienti (Monyei e Jenkins 2018).
Prezzi Interni del Carbonio: Alcune organizzazioni utilizzano prezzi ombra sulle emissioni di carbonio per rappresentare i costi ambientali nelle decisioni di allocazione del capitale tra progetti di IA. Se modellati in modo efficace, gli oneri teorici sulle impronte di carbonio dello sviluppo indirizzano i finanziamenti verso innovazioni efficienti piuttosto che solo verso guadagni di accuratezza.
Checklist per lo Sviluppo dell'Efficienza: Gruppi come AI Sustainability Coalition suggeriscono modelli di checklist volontari che evidenziano le scelte di progettazione del modello, le configurazioni hardware e altri fattori che gli architetti possono regolare per applicazione per limitare le emissioni. Le organizzazioni possono guidare il cambiamento radicando la sostenibilità come metrica di successo primaria insieme a precisione e costi.
Auditing Indipendente: Anche in assenza di mandati di divulgazione pubblica, le aziende specializzate in audit di sostenibilità tecnologica aiutano gli sviluppatori di IA a identificare gli

sprechi, creare roadmap di efficienza e confrontare i progressi tramite revisioni imparziali. Strutturare tali audit in procedure di governance interna o nel processo di approvvigionamento espande la responsabilità.

16.12.5 Considerazioni Globali

Mentre misurazione, restrizioni, incentivi e autoregolamentazione rappresentano potenziali meccanismi politici per promuovere la sostenibilità dell'IA, la frammentazione tra i regimi nazionali rischia di avere conseguenze indesiderate. Come per altri domini di politica tecnologica, la divergenza tra regioni deve essere gestita attentamente.

Ad esempio, a causa di preoccupazioni sulla privacy dei dati regionali, OpenAI ha impedito agli utenti europei di accedere al suo chatbot virale ChatGPT. Ciò è avvenuto dopo che la proposta di legge sull'IA dell'UE ha segnalato un approccio precauzionale, consentendo alla CE di vietare determinati usi dell'IA ad alto rischio e di imporre regole di trasparenza che creano incertezza per il rilascio di nuovi modelli. Tuttavia, sarebbe saggio mettere in guardia contro l'azione del regolatore in quanto potrebbe inavvertitamente limitare l'innovazione europea se i regimi con una regolamentazione più leggera attraggono più spesa e talenti per la ricerca sull'IA nel settore privato. Trovare un terreno comune è fondamentale.

I principi dell'OCSE sull'IA e i quadri delle Nazioni Unite sottolineano principi universalmente concordati che tutte le politiche nazionali dovrebbero sostenere: trasparenza, responsabilità, mitigazione dei "bias" [pregiudizi] e altro ancora. Incorporare in modo costruttivo la sostenibilità come principio fondamentale per un'IA responsabile all'interno di linee guida internazionali può motivare un'azione unitaria senza sacrificare la flessibilità tra sistemi legali divergenti. Evitare dinamiche di corsa al ribasso dipende da una cooperazione multilaterale illuminata.

16.13 Percezione e Coinvolgimento del Pubblico

Mentre l'attenzione della società e gli sforzi politici volti alla sostenibilità ambientale aumentano in tutto il mondo, cresce l'entusiasmo per l'utilizzo dell'intelligenza artificiale per aiutare ad affrontare le sfide ecologiche. Tuttavia, la comprensione e gli atteggiamenti del pubblico nei confronti del ruolo dei sistemi di IA nei contesti di sostenibilità devono ancora essere chiariti e sono offuscati da idee sbagliate. Da un lato, le persone sperano che algoritmi avanzati possano fornire nuove soluzioni per l'energia verde, il consumo responsabile, i percorsi di decarbonizzazione e la conservazione dell'ecosistema. Dall'altro, i timori sui rischi dell'IA incontrollata si insinuano anche nel dominio ambientale e minano il discorso costruttivo. Inoltre, una mancanza di consapevolezza pubblica su questioni chiave come la trasparenza nello sviluppo di strumenti di IA incentrati sulla sostenibilità e potenziali pregiudizi nei dati o nella modellazione minacciano anche di limitare la partecipazione inclusiva e degradare la fiducia del pubblico.

Affrontare priorità complesse e interdisciplinari come la sostenibilità ambientale richiede un coinvolgimento pubblico informato e sfumato e progressi responsabili nell'innovazione dell'IA. Il percorso da seguire richiede sforzi collaborativi attenti ed equi tra esperti in ML, climatologia, politica ambientale, scienze sociali e comunicazione. Mappare il panorama delle percezioni pubbliche, identificare le insidie e tracciare strategie per coltivare sistemi di IA comprensibili, accessibili e affidabili che puntino a priorità ecologiche condivise si rivelerà essenziale per realizzare obiettivi di sostenibilità. Questo terreno complesso giustifica un esame approfondito delle dinamiche socio-tecniche coinvolte.

16.13.1 Consapevolezza dell'IA

A maggio 2022, il Pew Research Center ha intervistato 5.101 adulti statunitensi, scoprendo che il 60% aveva sentito o letto "un po'" sull'IA mentre il 27% ne aveva sentito "molto", il che indica un discreto riconoscimento generale, ma probabilmente una comprensione limitata di

dettagli o applicazioni. Tuttavia, tra coloro che hanno una certa familiarità con l'IA, emergono preoccupazioni riguardo ai rischi di uso improprio dei dati personali secondo i termini concordati. Ciononostante, il 62% ritiene che l'IA potrebbe semplificare la vita moderna se applicata in modo responsabile. Tuttavia, una comprensione specifica dei contesti di sostenibilità deve ancora essere migliorata.
Gli studi che tentano di categorizzare i "sentiment" del discorso online rilevano una divisione quasi equa tra ottimismo e cautela riguardo all'implementazione dell'IA per obiettivi di sostenibilità. I fattori che guidano la positività includono le speranze di una migliore previsione dei cambiamenti ecologici utilizzando modelli di ML. La negatività nasce da una mancanza di fiducia negli algoritmi auto-supervisionati che evitano conseguenze indesiderate dovute a impatti umani imprevedibili su sistemi naturali complessi durante l'addestramento.
La convinzione pubblica più diffusa rimane che, mentre l'IA ha il potenziale per accelerare le soluzioni su questioni come la riduzione delle emissioni e la protezione della fauna selvatica, una salvaguardia inadeguata intorno a pregiudizi dei dati, punti ciechi etici e considerazioni sulla privacy potrebbero essere rischi più apprezzati se perseguiti con noncuranza, soprattutto su larga scala. Ciò porta a esitazione intorno al supporto incondizionato senza prove di uno sviluppo deliberato e guidato democraticamente.

16.13.2 Messaggistica

Gli sforzi ottimistici stanno evidenziando la promessa di sostenibilità dell'IA e sottolineano il potenziale del ML avanzato per accelerare radicalmente gli effetti di decarbonizzazione da reti intelligenti, app personalizzate di tracciamento del carbonio, ottimizzazioni automatizzate dell'efficienza degli edifici e analisi predittive che guidano gli sforzi di conservazione mirati. Una modellazione in tempo reale più completa di complessi cambiamenti climatici ed ecologici utilizzando algoritmi auto-miglioranti offre speranza per mitigare le perdite di biodiversità ed evitare gli scenari peggiori.
Tuttavia, prospettive cautelative, come i Principi di IA di Asilomar, mettono in dubbio se l'IA stessa potrebbe esacerbare le sfide della sostenibilità se vincolata in modo improprio. Le crescenti richieste di energia dei sistemi di elaborazione su larga scala e il training sempre più massiccio del modello di rete neurale sono in conflitto con le ambizioni di energia pulita. La mancanza di diversità negli input di dati o nelle priorità degli sviluppatori potrebbe sminuire le urgenti considerazioni di giustizia ambientale. L'impegno pubblico scettico a breve termine probabilmente dipende dalla necessità di salvaguardie percepibili contro i sistemi di intelligenza artificiale incontrollati che impazziscono nei processi ecologici fondamentali.
In sostanza, i "framing" polarizzati promuovono l'intelligenza artificiale come uno strumento indispensabile per la risoluzione dei problemi di sostenibilità, se indirizzata compassionevolmente verso le persone e il pianeta, oppure presentano l'IA come un amplificatore dei danni esistenti che dominano insidiosamente aspetti nascosti dei sistemi naturali centrali per tutta la vita. Superare tali impasse richiede di bilanciare discussioni oneste sui compromessi con visioni condivise per un progresso tecnologico equo e democraticamente governato che mira al ripristino.

16.13.3 Partecipazione Equa

Garantire una partecipazione e un accesso equi dovrebbe costituire la pietra angolare di qualsiasi iniziativa di sostenibilità con il potenziale per importanti impatti sociali. Questo principio si applica ugualmente ai sistemi di IA che mirano a obiettivi ambientali. Tuttavia, voci comunemente escluse come le comunità in prima linea, rurali o indigene e le generazioni future non presenti per il consenso potrebbero subire conseguenze sproporzionate dalle trasformazioni tecnologiche. Ad esempio, la Partnership on AI ha

lanciato eventi espressamente mirati al contributo delle comunità emarginate sull'implementazione responsabile dell'intelligenza artificiale.
Garantire un accesso e una partecipazione equi dovrebbe costituire la pietra angolare di qualsiasi iniziativa di sostenibilità con il potenziale per importanti impatti sociali, che si tratti di intelligenza artificiale o altro. Tuttavia, l'impegno inclusivo nell'intelligenza artificiale ambientale si basa in parte sulla disponibilità e sulla comprensione delle risorse informatiche fondamentali. Come sottolinea il recente rapporto OCSE sulla capacità di calcolo IA nazionale (Oecd 2023), molti paesi attualmente non dispongono di dati o piani strategici che mappino le esigenze per l'infrastruttura richiesta per alimentare i sistemi di IA. Questo punto cieco politico potrebbe limitare gli obiettivi economici ed esacerbare le barriere all'ingresso per le popolazioni emarginate. Il loro progetto sollecita lo sviluppo di strategie nazionali per la capacità di calcolo AI lungo dimensioni di capacità, accessibilità, pipeline di innovazione e resilienza per ancorare l'innovazione. L'archiviazione dei dati di base deve essere migliorata e le piattaforme di sviluppo dei modelli o l'hardware specializzato potrebbero inavvertitamente concentrare i progressi dell'AI nelle mani di gruppi selezionati. Pertanto, la pianificazione di un'espansione equilibrata delle risorse di calcolo AI fondamentali tramite iniziative politiche si collega direttamente alle speranze di una risoluzione dei problemi di sostenibilità democratizzata utilizzando strumenti ML equi e trasparenti.
L'idea chiave è che la partecipazione equa nei sistemi di intelligenza artificiale che affrontano le sfide ambientali si basa in parte sulla garanzia che la capacità di elaborazione e l'infrastruttura di base siano corrette, il che richiede una pianificazione politica proattiva da una prospettiva nazionale.

16.13.4 Trasparenza

Mentre le agenzie del settore pubblico e le aziende private si affrettano ad adottare strumenti di IA per aiutare ad affrontare le urgenti sfide ambientali, le richieste di trasparenza sullo sviluppo e la funzionalità di questi sistemi hanno iniziato ad amplificarsi. Le funzionalità di ML spiegabili e interpretabili diventano sempre più cruciali per creare fiducia nei modelli emergenti che mirano a guidare le conseguenti politiche di sostenibilità. Iniziative come il Montreal Carbon Pledge hanno riunito i leader della tecnologia per impegnarsi a pubblicare valutazioni di impatto prima di lanciare sistemi ambientali, come promesso di seguito:

> *"Come investitori istituzionali, dobbiamo agire nel migliore interesse a lungo termine dei nostri beneficiari. In questo ruolo fiduciario, i rischi di investimento a lungo termine sono associati alle emissioni di gas serra, ai cambiamenti climatici e alla regolamentazione del carbonio. Misurare la nostra impronta di carbonio è fondamentale per comprendere meglio, quantificare e gestire gli impatti, i rischi e le opportunità correlati al carbonio e ai cambiamenti climatici nei nostri investimenti. Pertanto, come primo passo, ci impegniamo a misurare e divulgare annualmente l'impronta di carbonio dei nostri investimenti per utilizzare queste informazioni per sviluppare una strategia di coinvolgimento e identificare e stabilire obiettivi di riduzione dell'impronta di carbonio." – Montréal Carbon Pledge*

Abbiamo bisogno di un impegno simile per la sostenibilità e la responsabilità dell'IA. L'accettazione diffusa e l'impatto delle soluzioni di sostenibilità dell'IA dipenderanno in parte dalla comunicazione deliberata di schemi di convalida, metriche e livelli di giudizio umano applicati prima dell'implementazione in tempo reale. Lavori come i Principi per l'IA spiegabile del NIST possono aiutare a promuovere la trasparenza nei sistemi di IA. Il National Institute of Standards and Technology (NIST) ha pubblicato un influente set di linee guida denominato "Principles for Explainable AI" [Principi per l'IA spiegabile] (Phillips et al. 2020). Questo

framework articola le best practice per la progettazione, la valutazione e l'implementazione di sistemi di IA responsabili con funzionalità trasparenti e interpretabili che creano comprensione e fiducia fondamentali per l'utente.
Delinea quattro principi fondamentali: in primo luogo, i sistemi di IA dovrebbero fornire spiegazioni contestualmente rilevanti che giustifichino il ragionamento alla base dei loro output alle parti interessate appropriate. In secondo luogo, queste spiegazioni di IA devono comunicare informazioni in modo significativo per il livello di comprensione appropriato del loro pubblico target. Il successivo è il principio di accuratezza, che stabilisce che le spiegazioni dovrebbero riflettere fedelmente il processo effettivo e la logica che informano i meccanismi interni di un modello di IA per generare output o raccomandazioni dati in base agli input. Infine, un principio di limiti di conoscenza obbliga le spiegazioni a chiarire i confini di un modello di IA nel catturare l'intera ampiezza della complessità, della varianza e delle incertezze del mondo reale all'interno di uno spazio problematico.
Nel complesso, questi principi NIST offrono ai professionisti e agli adottanti dell'IA una guida su considerazioni chiave sulla trasparenza, essenziali per sviluppare soluzioni accessibili che diano priorità all'autonomia e alla fiducia dell'utente piuttosto che semplicemente massimizzare le sole metriche di accuratezza predittiva. Man mano che l'IA avanza rapidamente in contesti sociali sensibili come sanità, finanza, occupazione e oltre, tali linee guida di progettazione incentrate sull'uomo continueranno a crescere in importanza per ancorare l'innovazione agli interessi pubblici.
Ciò si applica anche al dominio della capacità ambientale. Un'innovazione dell'IA responsabile e guidata democraticamente che mira a priorità ecologiche condivise dipende dal mantenimento della vigilanza pubblica, della comprensione e della supervisione su sistemi altrimenti opachi che assumono ruoli di primo piano nelle decisioni della società. Dare priorità a progetti di algoritmi spiegabili e pratiche di trasparenza radicale secondo standard globali può aiutare a sostenere la fiducia collettiva che questi strumenti migliorino piuttosto che mettere a repentaglio le speranze per un futuro guidato.

16.14 Direzioni e Sfide Future

Guardando al futuro, il ruolo dell'IA nella sostenibilità ambientale è destinato a crescere in modo ancora più significativo. Il potenziale dell'IA per guidare i progressi nell'energia rinnovabile, nella modellazione climatica, negli sforzi di conservazione e altro è immenso. Tuttavia, è una moneta a due facce, poiché dobbiamo superare diverse sfide e indirizzare i nostri sforzi verso uno sviluppo dell'IA sostenibile e responsabile.

16.14.1 Direzioni Future

Una delle direzioni chiave del futuro è lo sviluppo di modelli e algoritmi di intelligenza artificiale più efficienti dal punto di vista energetico. Ciò implica una ricerca e innovazione continue in aree come il "pruning" [potatura] dei modelli, la quantizzazione e l'uso di numeri a bassa precisione, nonché lo sviluppo dell'hardware per consentire la piena redditività di queste innovazioni. Inoltre, esaminiamo paradigmi di elaborazione alternativi che non si basano su architetture von-Neumann. Ulteriori informazioni su questo argomento sono disponibili nel capitolo sull'accelerazione hardware. L'obiettivo è creare sistemi di IA che offrano prestazioni elevate riducendo al minimo il consumo di energia e le emissioni di carbonio.
Un'altra direzione importante è l'integrazione di fonti di energia rinnovabili nell'infrastruttura di IA. Poiché i data center continuano a contribuire in modo significativo all'impronta di carbonio dell'IA, la transizione verso fonti di energia rinnovabili come l'energia solare ed eolica è fondamentale. Gli sviluppi nell'accumulo di energia sostenibile a lungo termine, come Ambri, uno spin-off del MIT, potrebbero consentire questa transizione. Ciò richiede

investimenti e collaborazioni significativi tra aziende tecnologiche, fornitori di energia e politici.

16.14.2 Sfide

Nonostante queste promettenti direzioni, devono essere affrontate diverse sfide. Una delle sfide principali è la necessità di standard e metodologie coerenti per misurare e segnalare l'impatto ambientale dell'IA. Questi metodi devono catturare la complessità dei cicli di vita dei modelli di IA e dell'hardware di sistema. Inoltre, sono necessarie infrastrutture IA e hardware di sistema efficienti e sostenibili dal punto di vista ambientale. Ciò è costituito da tre componenti:

1. Massimizzare l'utilizzo delle risorse di acceleratore e sistema.
2. Prolungare la durata di vita delle infrastrutture IA.
3. Progettare hardware di sistema tenendo presente l'impatto ambientale.

Dal lato software, dovremmo bilanciare la sperimentazione e il conseguente costo di training. Tecniche come la ricerca dell'architettura neurale e l'ottimizzazione degli iperparametri possono essere utilizzate per l'esplorazione dello spazio di progettazione. Tuttavia, queste sono spesso molto dispendiose in termini di risorse. Una sperimentazione efficiente può ridurre significativamente l'impatto ambientale. Successivamente, dovrebbero essere esplorati metodi per ridurre gli sforzi di training sprecati.

Per migliorare la qualità del modello, spesso ridimensioniamo il set di dati. Tuttavia, le maggiori risorse di sistema richieste per l'archiviazione e l'ingestione dei dati causate da questa scalabilità hanno un impatto ambientale significativo (C.-J. Wu et al. 2022). È importante comprendere a fondo la velocità con cui i dati perdono il loro valore predittivo e ideare strategie di campionamento dei dati.

Anche le lacune nei dati rappresentano una sfida significativa. Senza aziende e governi che condividono apertamente dati dettagliati e accurati sul consumo di energia, sulle emissioni di carbonio e su altri impatti ambientali, non è facile sviluppare strategie efficaci per un'IA sostenibile.

Infine, il ritmo rapido dello sviluppo dell'IA richiede un approccio agile alla politica imposta a questi sistemi. La politica dovrebbe garantire uno sviluppo sostenibile senza limitare l'innovazione. Ciò richiede che esperti in tutti i settori dell'IA, delle scienze ambientali, dell'energia e della politica lavorino insieme per raggiungere un futuro sostenibile.

16.15 Conclusione

Dobbiamo affrontare le considerazioni sulla sostenibilità man mano che l'intelligenza artificiale si espande rapidamente nei settori e nella società. L'intelligenza artificiale promette innovazioni rivoluzionarie, ma il suo impatto ambientale minaccia la sua crescita diffusa. Questo capitolo analizza molteplici aspetti, dall'energia e dalle emissioni agli impatti sui rifiuti e sulla biodiversità, che gli sviluppatori di intelligenza artificiale/apprendimento automatico devono valutare quando creano sistemi di IA responsabili.

Fondamentalmente, abbiamo bisogno di elevare la sostenibilità a priorità di progettazione primaria piuttosto che a un ripensamento. Tecniche come modelli ad alta efficienza energetica, data center alimentati da fonti rinnovabili e programmi di riciclaggio dell'hardware offrono soluzioni, ma l'impegno olistico rimane fondamentale. Abbiamo bisogno di standard in materia di trasparenza, contabilità del carbonio e divulgazioni della catena di fornitura per integrare i guadagni tecnici. Tuttavia, esempi come le pratiche di efficienza 4M di Google contenenti l'uso di energia ML evidenziano che possiamo far progredire l'intelligenza artificiale di pari passo con gli obiettivi ambientali con uno sforzo concertato. Raggiungiamo questo equilibrio armonioso facendo collaborare ricercatori,

aziende, regolatori e utenti in tutti i domini. L'obiettivo non è soluzioni perfette, ma un miglioramento continuo mentre integriamo l'IA in nuovi settori.

16.16 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- Transparency and Sustainability.
- Sustainability of TinyML.
- Model Cards for Transparency.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 16.1
- Esercizio 16.2

Laboratori

Oltre agli esercizi, offriamo laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di IA embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

17. IA Robusta

DALL·E 3 Prompt: Creare un'immagine che rappresenti un sistema di intelligenza artificiale avanzato simboleggiato da una rete neurale intricata e luminosa, profondamente annidata all'interno di una serie di scudi progressivamente più grandi e fortificati. Ogni strato di scudo rappresenta uno strato di difesa, che mostra la robustezza del sistema contro minacce esterne ed errori interni. La rete neurale, al centro di questa fortezza di scudi, irradia connessioni che simboleggiano la capacità di apprendimento e adattamento dell'intelligenza artificiale. Questa metafora visiva enfatizza non solo la sofisticatezza tecnologica dell'intelligenza artificiale, ma anche la sua resilienza e sicurezza, sullo sfondo di una sala server sicura e all'avanguardia, piena delle ultime innovazioni tecnologiche. L'immagine mira a trasmettere il concetto di massima protezione e resilienza nel campo dell'intelligenza artificiale.

Lo sviluppo di sistemi di apprendimento automatico robusti è diventato sempre più cruciale. Poiché questi sistemi vengono implementati in varie applicazioni critiche, dai veicoli autonomi alla diagnostica sanitaria, garantire la loro resilienza a guasti ed errori è fondamentale.

L'IA robusta, nel contesto di guasti hardware, guasti software ed errori, svolge un ruolo importante nel mantenimento dell'affidabilità, della sicurezza e delle prestazioni dei sistemi di apprendimento automatico. Affrontando le sfide poste da guasti hardware transitori, permanenti e intermittenti (Ahmadilivani et al. 2024), nonché bug, difetti di progettazione ed errori di implementazione nel software (H. Zhang 2008), le tecniche di intelligenza artificiale robuste consentono ai sistemi di apprendimento automatico di funzionare efficacemente anche in condizioni avverse.

Questo capitolo esplora i concetti fondamentali, le tecniche e gli strumenti per la creazione di sistemi di apprendimento automatico tolleranti ai guasti e resilienti agli errori. Consente a ricercatori e professionisti di sviluppare soluzioni di IA in grado di resistere alle complessità e alle incertezze degli ambienti del mondo reale.

Obiettivi dell'Apprendimento

- Comprendere l'importanza di sistemi di IA robusti e resilienti nelle applicazioni del mondo reale.
- Identificare e caratterizzare guasti hardware, guasti software e il loro impatto sui sistemi ML.
- Riconoscere e sviluppare strategie difensive contro le minacce poste da attacchi avversari, avvelenamento dei dati e cambiamenti nella distribuzione.
- Imparare tecniche per rilevare, mitigare e progettare sistemi ML tolleranti ai guasti.
- Acquisire familiarità con strumenti e framework per studiare e migliorare la resilienza del sistema ML durante l'intero ciclo di vita dello sviluppo dell'IA.

17.1 Introduzione

Per IA robusta si intende la capacità di un sistema di mantenere le proprie prestazioni e affidabilità anche in presenza di errori. Un sistema di apprendimento automatico robusto è progettato per essere tollerante ai guasti e resiliente agli errori, in grado di funzionare efficacemente anche in condizioni avverse.

Man mano che i sistemi ML diventano sempre più integrati in vari aspetti della nostra vita, dai servizi basati su cloud ai dispositivi edge e ai sistemi embedded, l'impatto dei guasti hardware e software sulle loro prestazioni e affidabilità diventa più significativo. In futuro, man mano che i sistemi ML diventano più complessi e vengono implementati in applicazioni ancora più critiche, la necessità di progetti robusti e tolleranti ai guasti sarà fondamentale.

Si prevede che i sistemi ML svolgeranno ruoli cruciali nei veicoli autonomi, nelle città intelligenti, nell'assistenza sanitaria e nei domini dell'automazione industriale. In questi domini, le conseguenze dei guasti hardware o software possono essere gravi, potenzialmente causa di perdita di vite umane, danni economici o danni ambientali.

I ricercatori e gli ingegneri devono concentrarsi sullo sviluppo di tecniche avanzate per il rilevamento, l'isolamento e il ripristino dei guasti per mitigare questi rischi e garantire il funzionamento affidabile dei futuri sistemi ML.

Questo capitolo si concentrerà in modo specifico su tre categorie principali di guasti ed errori che possono influire sulla robustezza dei sistemi ML: guasti hardware, guasti software ed errori umani.

- **Guasti Hardware:** Guasti transitori, permanenti e intermittenti possono influire sui componenti hardware di un sistema ML, corrompendo i calcoli e degradando le prestazioni.
- **Robustezza del Modello:** I modelli ML possono essere vulnerabili ad attacchi avversari, avvelenamento dei dati e cambiamenti di distribuzione, che possono indurre classificazioni errate mirate, alterare il comportamento appreso del modello o compromettere l'integrità e l'affidabilità del sistema.
- **Guasti software:** Bug, difetti di progettazione ed errori di implementazione nei componenti software, come algoritmi, librerie e framework, possono propagare errori e introdurre vulnerabilità.

Le sfide e gli approcci specifici per ottenere la robustezza possono variare a seconda della scala e dei vincoli del sistema ML. I sistemi di cloud computing o data center su larga scala possono concentrarsi sulla tolleranza ai guasti e sulla resilienza tramite ridondanza, elaborazione distribuita e tecniche avanzate di rilevamento e correzione degli errori. Al contrario, i dispositivi edge con risorse limitate o i sistemi embedded affrontano sfide uniche a causa della potenza di calcolo, della memoria e delle risorse energetiche limitate.

Indipendentemente dalla scala e dai vincoli, le caratteristiche chiave di un sistema ML robusto includono tolleranza ai guasti, resilienza agli errori e mantenimento delle prestazioni. Comprendendo e affrontando le sfide multiformi alla robustezza, possiamo sviluppare sistemi ML affidabili e sicuri in grado di navigare nelle complessità degli ambienti del mondo reale.

Questo capitolo non riguarda solo l'esplorazione di strumenti, framework e tecniche dei sistemi ML per rilevare e mitigare guasti, attacchi e cambiamenti durante la distribuzione. Si tratta di sottolineare il ruolo cruciale di ognuno di nel dare priorità alla resilienza durante tutto il ciclo di vita dello sviluppo dell'IA, dalla raccolta dati e dall'addestramento del modello all'implementazione e al monitoraggio. Affrontando in modo proattivo le sfide alla robustezza, possiamo sbloccare il pieno potenziale delle tecnologie ML garantendone al contempo un'implementazione sicura, affidabile e responsabile nelle applicazioni del mondo reale.

Mentre l'IA continua a plasmare il nostro futuro, il potenziale delle tecnologie ML è immenso. Ma è solo quando creiamo sistemi resilienti in grado di resistere alle sfide del mondo reale che possiamo davvero sfruttare questo potenziale. Questo è un fattore determinante per il successo e l'impatto sociale di questa tecnologia trasformativa ed è alla nostra portata.

17.2 Esempi del mondo reale

Ecco alcuni esempi reali di casi in cui guasti nell'hardware o nel software hanno causato problemi importanti nei sistemi ML in ambienti cloud, edge ed embedded:

17.2.1 Cloud

Nel febbraio 2017, Amazon Web Services (AWS) ha subito un'interruzione significativa a causa di un errore umano durante la manutenzione. Un tecnico ha inserito inavvertitamente un comando errato, causando la disconnessione di molti server. Questa interruzione ha interrotto molti servizi AWS, tra cui l'assistente basato sull'intelligenza artificiale di Amazon, Alexa. Di conseguenza, i dispositivi basati su Alexa, come Amazon Echo e prodotti di terze parti che utilizzano Alexa Voice Service, non hanno potuto rispondere alle richieste degli utenti per diverse ore. Questo incidente evidenzia il potenziale impatto degli errori umani sui sistemi ML basati su cloud e la necessità di procedure di manutenzione robuste e meccanismi di sicurezza.

In un altro esempio (Vangal et al. 2021), Facebook ha riscontrato un problema di "silent data corruption (SDC)" [corruzione silenziosa dei dati] all'interno della sua infrastruttura di query distribuita, come mostrato in Figura 17.1. L'infrastruttura di Facebook include un sistema di query che preleva ed esegue query SQL e simili a SQL su più set di dati utilizzando framework come Presto, Hive e Spark. Una delle applicazioni che ha utilizzato questa infrastruttura di query è stata un'applicazione di compressione per ridurre l'ingombro degli archivi dati. In questa applicazione di compressione, i file venivano compressi quando non venivano letti e decompressi quando veniva effettuata una richiesta di lettura. Prima della decompressione, la dimensione del file veniva controllata per assicurarsi che fosse maggiore di zero, indicando un file compresso valido con contenuti.

Figura 17.1: Corruzione silenzioso dei dati nelle applicazioni di database. Fonte: Facebook

Tuttavia, in un caso, quando la dimensione del file veniva calcolata per un file valido di dimensioni diverse da zero, l'algoritmo di decompressione ha richiamato una funzione di potenza dalla libreria Scala. Inaspettatamente, la funzione Scala ha restituito un valore di dimensione zero per il file nonostante avesse una dimensione decompressa nota diversa da zero. Di conseguenza, la decompressione non è stata eseguita e il file non è stato scritto nel database di output. Questo problema si è manifestato sporadicamente, con alcune occorrenze dello stesso calcolo della dimensione del file che restituivano il valore corretto diverso da zero.

L'impatto di questa corruzione silenziosa dei dati è stato significativo, portando a file mancanti e dati errati nel database di output. L'applicazione che si basava sui file decompressi ha fallito a causa delle incongruenze dei dati. Nel caso di studio presentato nel documento, l'infrastruttura di Facebook, che consiste in centinaia di migliaia di server che gestiscono miliardi di richieste al giorno dalla loro enorme base di utenti, ha riscontrato un problema di corruzione silenziosa dei dati. Il sistema interessato elaborava query utente, caricamenti di immagini e contenuti multimediali, che richiedevano un'esecuzione rapida, affidabile e sicura.

Questo caso di studio illustra come la corruzione silenziosa dei dati può propagarsi attraverso più strati di uno stack applicativo, causando perdita di dati e guasti delle applicazioni in un sistema distribuito su larga scala. La natura intermittente del problema e la mancanza di messaggi di errore espliciti lo hanno reso particolarmente difficile da diagnosticare e risolvere. Ma questo non è limitato solo a Meta, anche altre aziende come Google che gestiscono ipercomputer IA affrontano questi problemi. Figura 17.2 Jeff Dean, Chief Scientist presso Google DeepMind e Google Research, parla degli SDC e del loro impatto sui sistemi di apprendimento automatico.

Figura 17.2: Gli errori "Silent data corruption (SDC)" sono un problema importante per gli ipercomputer di IA. Fonte: Jeff Dean at MLSys 2024, Keynote (Google)

17.2.2 Edge

Per quanto riguarda esempi di guasti ed errori nei sistemi edge ML, un'area che ha ricevuto notevole attenzione è il dominio delle auto a guida autonoma. I veicoli a guida autonoma si basano in larga misura su algoritmi di apprendimento automatico per la percezione, il processo decisionale e il controllo, rendendoli particolarmente sensibili all'impatto di guasti hardware e software. Negli ultimi anni, diversi incidenti di alto profilo che hanno coinvolto veicoli autonomi hanno evidenziato le sfide e i rischi associati all'implementazione di questi sistemi in ambienti reali.

A maggio 2016, si è verificato un incidente mortale quando una Tesla Model S con pilota automatico si è schiantata contro un autoarticolato bianco che attraversava l'autostrada. Il sistema Autopilot, che si basava su algoritmi di visione artificiale e apprendimento automatico, non è riuscito a riconoscere il rimorchio bianco sullo sfondo di un cielo luminoso. Il conducente, che secondo quanto riferito stava guardando un film al momento dell'incidente, non è intervenuto in tempo e il veicolo è entrato in collisione con il rimorchio a tutta velocità. Questo incidente ha sollevato preoccupazioni sui limiti dei sistemi di percezione basati sull'intelligenza artificiale e sulla necessità di solidi meccanismi di sicurezza nei veicoli autonomi. Ha inoltre evidenziato l'importanza della consapevolezza del conducente e la necessità di linee guida chiare sull'uso delle funzionalità di guida semi-autonoma, come mostrato in Figura 17.3.

Figura 17.3: Tesla nell'incidente mortale in California era in modalità Autopilot. Fonte: BBC News

A marzo 2018, un veicolo di prova a guida autonoma di Uber ha investito e ucciso un pedone che attraversava la strada a Tempe, in Arizona. L'incidente è stato causato da un difetto software nel sistema di riconoscimento degli oggetti del veicolo, che non è riuscito a

identificare i pedoni in modo appropriato per evitarli come ostacoli. L'autista di sicurezza, che avrebbe dovuto monitorare il funzionamento del veicolo e intervenire se necessario, è stato trovato distratto durante l'incidente. Questo incidente ha portato ad un'ampia revisione del programma di guida autonoma di Uber e ha sollevato dubbi sulla prontezza della tecnologia dei veicoli autonomi per le strade pubbliche. Ha inoltre sottolineato la necessità di rigorosi test, convalide e misure di sicurezza nello sviluppo e nell'implementazione di sistemi di guida autonoma basati sull'intelligenza artificiale.
Nel 2021, Tesla ha dovuto affrontare un controllo più rigoroso a seguito di diversi incidenti che hanno coinvolto veicoli in modalità Autopilot. Alcuni di questi incidenti sono stati attribuiti a problemi con la capacità del sistema Autopilot di rilevare e rispondere a determinate situazioni stradali, come veicoli di emergenza fermi o ostacoli sulla strada. Ad esempio, nell'aprile 2021, una Tesla Model S si è schiantata contro un albero in Texas, uccidendo due passeggeri. I primi rapporti suggerivano che nessuno si trovasse al posto di guida al momento dell'incidente, sollevando interrogativi sull'uso e il potenziale uso improprio delle funzionalità Autopilot. Questi incidenti evidenziano le sfide in corso nello sviluppo di sistemi di guida autonoma affidabili e robusti e la necessità di normative chiare e di istruzione dei consumatori in merito alle capacità e ai limiti di queste tecnologie.

17.2.3 Embedded

I sistemi embedded, che spesso operano in ambienti con risorse limitate e applicazioni critiche per la sicurezza, hanno da tempo dovuto affrontare sfide legate a guasti hardware e software. Poiché le tecnologie di IA e apprendimento automatico sono sempre più integrate in questi sistemi, il potenziale di guasti ed errori assume nuove dimensioni, con l'aggiunta di complessità degli algoritmi di IA e la natura critica delle applicazioni in cui vengono distribuiti. Consideriamo alcuni esempi, a partire dall'esplorazione dello spazio. La missione Mars Polar Lander della NASA nel 1999 ha subito un guasto catastrofico a causa di un errore software nel sistema di rilevamento dell'atterraggio (Figura 17.4). Il software di bordo della navicella spaziale ha interpretato erroneamente il rumore proveniente dall'apertura delle sue gambe di atterraggio come un segnale di atterraggio sulla superficie marziana. Di conseguenza, la navicella ha spento prematuramente i suoi motori, causando lo schianto sulla superficie. Questo incidente evidenzia l'importanza critica di una progettazione software solida e di test approfonditi nei sistemi embedded, in particolare quelli che operano in ambienti remoti e ostili. Poiché le capacità di IA sono integrate nelle future missioni spaziali, garantire l'affidabilità e la tolleranza ai guasti di questi sistemi sarà fondamentale per il successo della missione.

Figura 17.4: La missione fallita della NASA Mars Polar Lander nel 1999 è costata oltre $200M. Fonte: SlashGear

Tornando sulla Terra, nel 2015, un Boeing 787 Dreamliner ha subito un arresto elettrico completo durante un volo a causa di un bug del software nelle sue unità di controllo del generatore. Questo incidente sottolinea come i guasti software possano avere gravi conseguenze nei sistemi integrati complessi come quelli degli aeromobili. Poiché le tecnologie di IA sono sempre più applicate all'aviazione, come nei sistemi di volo autonomi e nella manutenzione predittiva, garantire la robustezza e l'affidabilità di questi sistemi sarà fondamentale per la sicurezza dei passeggeri.

> *"Se le quattro unità di controllo del generatore principale (associate ai generatori montati sul motore) fossero accese contemporaneamente, dopo 248 giorni di alimentazione continua, tutte e quattro le GCU entrerebbero in modalità fail-safe contemporaneamente, con conseguente perdita di tutta l'alimentazione elettrica CA indipendentemente dalla fase di volo." – Direttiva della Federal Aviation Administration (2015)*

Poiché le capacità di IA si integrano sempre di più nei sistemi embedded, il potenziale di guasti ed errori diventa più complesso e grave. Si immagini un pacemaker intelligente che ha un improvviso problema tecnico. Un paziente potrebbe morire a causa di tale effetto. Pertanto, gli algoritmi AI, come quelli utilizzati per la percezione, il processo decisionale e il controllo, introducono nuove fonti di potenziali guasti, come problemi relativi ai dati, incertezze del modello e comportamenti inaspettati nei casi limite. Inoltre, la natura opaca di alcuni modelli di IA può rendere difficile identificare e diagnosticare i guasti quando si verificano.

17.3 Guasti Hardware

I guasti hardware rappresentano una sfida significativa nei sistemi informatici, inclusi i sistemi tradizionali e ML. Questi guasti si verificano quando componenti fisici, come processori, moduli di memoria, dispositivi di archiviazione o interconnessioni, funzionano male o si comportano in modo anomalo. I guasti hardware possono causare calcoli errati, danneggiamento dei dati, crash del sistema o guasti completi del sistema, compromettendo l'integrità e l'affidabilità dei calcoli eseguiti (S. Jha et al. 2019). Un guasto completo del sistema si riferisce a una situazione in cui l'intero sistema informatico diventa non reattivo o inutilizzabile a causa di un malfunzionamento hardware critico. Questo tipo di guasto è il più grave, poiché rende il sistema inutilizzabile e può portare alla perdita o al danneggiamento dei dati, richiedendo un intervento manuale per riparare o sostituire i componenti difettosi.
Comprendere la tassonomia dei guasti hardware è essenziale per chiunque lavori con sistemi informatici, in particolare nel contesto dei sistemi ML. I sistemi ML si basano su architetture hardware complesse e calcoli su larga scala per addestrare e distribuire modelli che apprendono dai dati e fanno previsioni o decisioni intelligenti. Tuttavia, i guasti hardware possono introdurre errori e incongruenze nella pipeline MLOps, influenzando l'accuratezza, la robustezza e l'affidabilità dei modelli addestrati (G. Li et al. 2017).
Conoscere i diversi tipi di guasti hardware, i loro meccanismi e il loro potenziale impatto sul comportamento del sistema è fondamentale per sviluppare strategie efficaci per rilevarli, mitigarli e ripristinarli. Questa conoscenza è necessaria per progettare sistemi di elaborazione tolleranti ai guasti, implementare algoritmi ML robusti e garantire l'affidabilità complessiva delle applicazioni basate su ML.
Le sezioni seguenti esploreranno le tre categorie principali di guasti hardware: transitori, permanenti e intermittenti. Discuteremo le loro definizioni, caratteristiche, cause, meccanismi ed esempi di come si manifestano nei sistemi di elaborazione. Tratteremo anche tecniche di rilevamento e mitigazione specifiche per ogni tipo di guasto.

- **Guasti Transitori:** I guasti transitori sono temporanei e non ricorrenti. Sono spesso causati da fattori esterni come raggi cosmici, interferenze elettromagnetiche o fluttuazioni di potenza. Un esempio comune di guasto transitorio è un bit flip, in cui un singolo bit in una posizione di memoria o registro cambia il suo valore in modo imprevisto. I guasti transitori possono causare calcoli errati o corruzione dei dati, ma non causano danni permanenti all'hardware.
- **Guasti permanenti:** I guasti permanenti, chiamati anche errori hard, sono irreversibili e persistono nel tempo. Sono in genere causati da difetti fisici o usura dei componenti hardware. Esempi di guasti permanenti includono guasti bloccati, in cui un bit o un segnale è impostato in modo permanente su un valore specifico (ad esempio, sempre 0 o sempre 1) e guasti del dispositivo, come un processore malfunzionante o un modulo di memoria danneggiato. I guasti permanenti possono causare un guasto completo del sistema o un significativo degrado delle prestazioni.
- **Guasti Intermittenti:** I guasti intermittenti sono guasti ricorrenti che compaiono e scompaiono in modo intermittente. Condizioni hardware instabili, come connessioni allentate, componenti obsoleti o difetti di fabbricazione, spesso ne sono la causa. I guasti intermittenti possono essere difficili da diagnosticare e riprodurre perché possono verificarsi sporadicamente e in condizioni specifiche. Esempi includono cortocircuiti intermittenti o problemi di resistenza dei contatti. I guasti intermittenti possono portare a un comportamento imprevedibile del sistema e a errori intermittenti.

Alla fine di questa discussione, i lettori avranno una solida comprensione della tassonomia dei guasti e della sua rilevanza per i sistemi di elaborazione e ML tradizionali. Questa base li aiuterà a prendere decisioni informate durante la progettazione, l'implementazione e la distribuzione di soluzioni tolleranti ai guasti, migliorando l'affidabilità e la credibilità dei loro sistemi di elaborazione e delle applicazioni ML.

17.3.1 Guasti Transitori

I guasti transitori nell'hardware possono manifestarsi in varie forme, ciascuna con le sue caratteristiche e cause uniche. Questi guasti sono di natura temporanea e non causano danni permanenti ai componenti hardware.

17.3.1.1 Definizione e Caratteristiche

Alcuni dei tipi comuni di guasti transitori includono Single Event Upset (SEU) causati da radiazioni ionizzanti, fluttuazioni di tensione (V. J. Reddi e Gupta 2013) dovute a rumore dell'alimentatore o interferenze elettromagnetiche, "Electromagnetic Interference (EMI)" indotte da campi elettromagnetici esterni, "Electrostatic Discharge (ESD)" risultanti da un improvviso flusso di elettricità statica, diafonia causata da accoppiamento di segnali involontari, rimbalzo di massa innescato dalla commutazione simultanea di più uscite, violazioni di temporizzazione dovute a violazioni dei vincoli di temporizzazione del segnale ed errori soft nella logica combinatoria che influenzano l'uscita dei circuiti logici (Mukherjee, Emer, e Reinhardt 2005). Comprendere questi diversi tipi di guasti transitori è fondamentale per progettare sistemi hardware robusti e resilienti che possano mitigarne l'impatto e garantire un funzionamento affidabile.

Tutti questi guasti transitori sono caratterizzati dalla loro breve durata e dalla loro natura non permanente. Non persistono né lasciano alcun impatto duraturo sull'hardware. Tuttavia, possono comunque portare a calcoli errati, corruzione dei dati o comportamento scorretto del sistema se non gestiti correttamente.

17.3.1.2 Cause di Guasti Transitori

I guasti transitori possono essere attribuiti a vari fattori esterni. Una causa comune sono i raggi cosmici, particelle ad alta energia provenienti dallo spazio. Quando queste particelle colpiscono aree sensibili dell'hardware, come celle di memoria o transistor, possono indurre disturbi di carica che alterano i dati memorizzati o trasmessi. Ciò è illustrato in Figura 17.5. Un'altra causa di guasti transitori è l'electromagnetic interference (EMI) [interferenza elettromagnetica] da dispositivi vicini o fluttuazioni di potenza. L'EMI può accoppiarsi con i circuiti e causare picchi di tensione o glitch che interrompono temporaneamente il normale funzionamento dell'hardware.

Figura 17.5: Meccanismo di Occorrenza di Guasti Transitori Hardware. Fonte: NTT

17.3.1.3 Meccanismi di Guasti Transitori

I guasti transitori possono manifestarsi attraverso meccanismi diversi a seconda del componente hardware interessato. Nei dispositivi di memoria come DRAM o SRAM, i guasti transitori spesso portano a inversioni di bit, in cui un singolo bit cambia il suo valore da 0 a 1 o viceversa. Ciò può corrompere i dati o le istruzioni archiviati. Nei circuiti logici, i guasti transitori possono causare glitch o picchi di tensione che si propagano attraverso la logica combinatoria, con conseguenti output o segnali di controllo errati. I guasti transitori possono anche influenzare i canali di comunicazione, causando errori di bit o perdite di pacchetti durante la trasmissione dei dati.

17.3.1.4 Impatto sui Sistemi ML

Un esempio comune di guasto transitorio è un'inversione di bit nella memoria principale. Se una struttura dati importante o un'istruzione critica viene archiviata nella posizione di

memoria interessata, può portare a calcoli errati o a un comportamento errato del programma. Se si verifica un guasto transitorio nella memoria che archivia i pesi o i gradienti del modello. Ad esempio, un bit flip nella memoria che memorizza un contatore di loop può causare l'esecuzione indefinita del loop o la sua terminazione prematura. Errori transitori nei registri di controllo o nei bit di flag possono alterare il flusso di esecuzione del programma, causando salti imprevisti o decisioni di diramazione errate. Nei sistemi di comunicazione, gli errori transitori possono danneggiare i pacchetti di dati trasmessi, causando ritrasmissioni o perdita di dati.

Nei sistemi ML, gli errori transitori possono avere implicazioni significative durante la fase di training (Y. He et al. 2023). Il training ML comporta calcoli iterativi e aggiornamenti dei parametri del modello basati su grandi set di dati. Se si verifica un errore transitorio nella memoria dei pesi o dei gradienti del modello, può causare aggiornamenti errati e compromettere la convergenza e l'accuratezza del processo di training. Figura 17.6 mostra un esempio concreto tratto dalla flotta di produzione di Google, in cui un'anomalia SDC ha causato una differenza significativa nella norma del gradiente.

Figura 17.6: SDC nella fase di training ML determina anomalie nella norma del gradiente. Fonte: Jeff Dean, MLSys 2024 Keynote (Google)

Ad esempio, un'inversione di bit nella matrice dei pesi di una rete neurale può far sì che il modello apprenda pattern o associazioni errati, con conseguente peggioramento delle prestazioni (Wan et al. 2021). Errori transitori nella pipeline dei dati, come la corruzione dei campioni di training o delle etichette, possono anche introdurre rumore e influire sulla qualità del modello appreso.

Durante la fase di inferenza, gli errori transitori possono influire sull'affidabilità e l'attendibilità delle previsioni ML. Se si verifica un errore transitorio nella memoria dei parametri del modello addestrato o nel calcolo dei risultati dell'inferenza, può portare a previsioni errate o incoerenti. Ad esempio, un'inversione di bit nei valori di attivazione di una

rete neurale può alterare l'output finale di classificazione o regressione (Mahmoud et al. 2020).
Nelle applicazioni "safety-critical", come i veicoli autonomi o la diagnosi medica, i guasti transitori durante l'inferenza possono avere gravi conseguenze, portando a decisioni o azioni errate (G. Li et al. 2017; S. Jha et al. 2019). Garantire la resilienza dei sistemi ML contro i guasti transitori è fondamentale per mantenere l'integrità e l'affidabilità delle previsioni.
All'altro estremo, in ambienti con risorse limitate come TinyML, le "Binarized Neural Networks [BNNs]" [reti neurali binarizzate] (Courbariaux et al. 2016) sono emerse come una soluzione promettente. Le BNN rappresentano pesi di rete in precisione a bit singolo, offrendo efficienza computazionale e tempi di inferenza più rapidi. Tuttavia, questa rappresentazione binaria rende le BNN fragili agli errori di inversione di bit sui pesi della rete. Ad esempio, lavori precedenti (Aygun, Gunes, e De Vleeschouwer 2021) hanno dimostrato che un'architettura BNN a due strati nascosti per un'attività semplice come la classificazione MNIST subisce un degrado delle prestazioni dal 98% di accuratezza del test al 70% quando vengono inseriti errori soft di inversione di bit casuali tramite pesi del modello con una probabilità del 10%.
Per affrontare tali problemi è necessario considerare tecniche di training "flip-aware" o sfruttare paradigmi di elaborazione emergenti (ad esempio, elaborazione stocastica) per migliorare la tolleranza ai guasti e la robustezza, di cui parleremo in Sezione 17.3.4. Le direzioni di ricerca future mirano a sviluppare architetture ibride, nuove funzioni di attivazione e funzioni di perdita su misura per colmare il divario di accuratezza rispetto ai modelli a precisione completa mantenendo al contempo la loro efficienza computazionale.

17.3.2 Guasti Permanenti

I guasti permanenti sono difetti hardware che persistono e causano danni irreversibili ai componenti interessati. Questi guasti sono caratterizzati dalla loro natura persistente e richiedono la riparazione o la sostituzione dell'hardware difettoso per ripristinare la normale funzionalità del sistema.

17.3.2.1 Definizione e Caratteristiche

I guasti permanenti sono difetti hardware che causano malfunzionamenti persistenti e irreversibili nei componenti interessati. Il componente difettoso rimane non operativo finché un guasto permanente non viene riparato o sostituito. Questi guasti sono caratterizzati dalla loro natura coerente e riproducibile, il che significa che il comportamento difettoso viene osservato ogni volta che il componente interessato viene utilizzato. I guasti permanenti possono avere un impatto su vari componenti hardware, come processori, moduli di memoria, dispositivi di archiviazione o interconnessioni, causando crash del sistema, danneggiamento dei dati o guasto completo del sistema.
Un esempio notevole di guasto permanente è il bug Intel FDIV, scoperto nel 1994. Il bug FDIV era un difetto in alcune unità di divisione a virgola mobile (FDIV) dei processori Intel Pentium. Il bug causava risultati errati per specifiche operazioni di divisione, portando a calcoli imprecisi.
Il bug FDIV si è verificato a causa di un errore nella tabella di ricerca utilizzata dall'unità di divisione. In rari casi, il processore recuperava un valore errato dalla tabella di ricerca, con un risultato leggermente meno preciso del previsto. Ad esempio, Figura 17.7 mostra una frazione 4195835/3145727 tracciata su un processore Pentium con l'errore permanente FDIV. Le regioni triangolari sono quelle in cui si sono verificati calcoli errati. Idealmente, tutti i valori corretti verrebbero arrotondati a 1,3338, ma i risultati errati mostrano 1,3337, indicando un errore nella quinta cifra.
Sebbene l'errore fosse piccolo, poteva accumularsi su molte operazioni di divisione, portando a significative imprecisioni nei calcoli matematici. L'impatto del bug FDIV era

significativo, soprattutto per le applicazioni che si basavano in modo massiccio sulla divisione precisa in virgola mobile, come simulazioni scientifiche, calcoli finanziari e progettazione assistita da computer. Il bug ha portato a risultati errati, che potrebbero avere gravi conseguenze in settori come la finanza o l'ingegneria.

Figura 17.7: Processore Intel Pentium con errore permanente FDIV. Le regioni triangolari sono quelle in cui si sono verificati calcoli errati. Fonte: Byte Magazine

Il bug Intel FDIV è un monito per il potenziale impatto di guasti permanenti sui sistemi ML. Nel contesto del ML, guasti permanenti nei componenti hardware possono portare a calcoli errati, influenzando l'accuratezza e l'affidabilità dei modelli. Ad esempio, se un sistema ML si basa su un processore con un'unità a virgola mobile difettosa, simile al bug Intel FDIV, potrebbe introdurre errori nei calcoli eseguiti durante l'addestramento o l'inferenza.
Questi errori possono propagarsi attraverso il modello, portando a previsioni imprecise o apprendimento distorto. Nelle applicazioni in cui il ML viene utilizzato per attività critiche, come la guida autonoma, la diagnosi medica o le previsioni finanziarie, le conseguenze di calcoli errati dovuti a guasti permanenti possono essere gravi.
È fondamentale che i professionisti del ML siano consapevoli del potenziale impatto dei guasti permanenti e incorporino tecniche di tolleranza ai guasti, come ridondanza hardware, meccanismi di rilevamento e correzione degli errori e progettazione di algoritmi robusti, per mitigare i rischi associati a questi guasti. Inoltre, test approfonditi e convalida dei componenti hardware ML possono aiutare a identificare e risolvere i guasti permanenti prima che influiscano sulle prestazioni e l'affidabilità del sistema.

17.3.2.2 Cause dei Guasti Permanenti

I guasti permanenti possono derivare da diverse cause, tra cui difetti di fabbricazione e meccanismi di usura. I difetti di fabbricazione sono difetti intrinseci introdotti durante il processo di fabbricazione dei componenti hardware. Questi difetti includono incisione impropria, doping non corretto o contaminazione, che portano a componenti non funzionali o parzialmente funzionali.
D'altro canto, i meccanismi di usura si verificano nel tempo man mano che i componenti hardware sono sottoposti a un uso prolungato e a stress. Fattori come elettromigrazione, rottura dell'ossido o stress termico possono causare una graduale degradazione dei componenti, portando infine a guasti permanenti.

17.3.2.3 Meccanismi dei Guasti Permanenti

I guasti permanenti possono manifestarsi attraverso vari meccanismi, a seconda della natura e della posizione del guasto. Gli "Stuck-at fault" [guasti bloccati] (Seong et al. 2010) sono guasti permanenti comuni in cui un segnale o una cella di memoria rimane fissata a un valore particolare (0 o 1) indipendentemente dagli input, come illustrato in Figura 17.8.

Figura 17.8: Modello di Guasto Bloccato nei Circuiti Digitali. Fonte: Accendo Reliability

I guasti bloccati possono verificarsi in porte logiche, celle di memoria o interconnessioni, causando calcoli errati o corruzione dei dati. Un altro meccanismo sono i guasti del dispositivo, in cui un componente, come un transistor o una cella di memoria, cessa completamente di funzionare. Ciò può essere dovuto a difetti di fabbricazione o grave usura. I guasti di "bridging" si verificano quando due o più linee di segnale sono collegate involontariamente, causando cortocircuiti o un comportamento logico errato.

Oltre ai guasti stuck-at, ci sono diversi altri tipi di guasti permanenti che possono influenzare i circuiti digitali e che possono avere un impatto su un sistema ML. I guasti di ritardo possono causare il superamento del limite specificato del ritardo di propagazione di un segnale, portando a violazioni di temporizzazione. I guasti di interconnessione, come guasti aperti (fili rotti), guasti resistivi (resistenza aumentata) o guasti capacitivi (capacità aumentata), possono causare problemi di integrità del segnale o violazioni di temporizzazione. Le celle di memoria possono anche subire vari guasti, tra cui guasti di transizione (impossibilità di cambiare stato), guasti di accoppiamento (interferenza tra celle adiacenti) e guasti sensibili al pattern di vicinato (guasti che dipendono dai valori delle celle vicine). Altri guasti permanenti possono verificarsi nella rete di alimentazione o nella rete di distribuzione del clock, influenzando la funzionalità e la temporizzazione del circuito.

17.3.2.4 Impatto sui Sistemi ML

I guasti permanenti possono influire gravemente sul comportamento e l'affidabilità dei sistemi di elaborazione. Ad esempio, un guasto nell'unità logica aritmetica (ALU) di un processore può causare calcoli errati, portando a risultati errati o crash del sistema. Un guasto permanente in un modulo di memoria, in una specifica cella di memoria, può danneggiare i dati archiviati, causando la perdita di dati o un comportamento errato del programma. Nei dispositivi di archiviazione, guasti permanenti come settori danneggiati o guasti del dispositivo possono causare l'inaccessibilità dei dati o la perdita completa delle informazioni archiviate. I guasti permanenti di interconnessione possono interrompere i canali di comunicazione, causando il danneggiamento dei dati o il blocco del sistema.

I guasti permanenti possono influire significativamente sui sistemi ML durante le fasi di addestramento e inferenza. Durante l'addestramento, guasti permanenti nelle unità di elaborazione o nella memoria possono causare calcoli errati, con conseguenti modelli danneggiati o non ottimali (Y. He et al. 2023). Inoltre, i guasti nei dispositivi di archiviazione

possono corrompere i dati di training o i parametri del modello archiviati, causando la perdita di dati o incongruenze del modello (Y. He et al. 2023).
Durante l'inferenza, i guasti permanenti possono influire sull'affidabilità e la correttezza delle previsioni ML. I guasti nelle unità di elaborazione possono produrre risultati errati o causare guasti del sistema, mentre i guasti nella memoria che archivia i parametri del modello possono portare all'utilizzo di modelli corrotti o obsoleti per l'inferenza (J. J. Zhang et al. 2018).
Per mitigare l'impatto dei guasti permanenti nei sistemi ML, devono essere impiegate tecniche di tolleranza ai guasti sia a livello hardware che software. La ridondanza hardware, come la duplicazione di componenti critici o l'utilizzo di codici di correzione degli errori (J. Kim, Sullivan, e Erez 2015), può aiutare a rilevare e ripristinare i guasti permanenti. Le tecniche software, come i meccanismi di checkpoint e riavvio (Egwutuoha et al. 2013), possono consentire al sistema di recuperare da guasti permanenti tornando a uno stato salvato in precedenza. Il monitoraggio, il test e la manutenzione regolari dei sistemi ML possono aiutare a identificare e sostituire i componenti difettosi prima che causino interruzioni significative.
Progettare sistemi ML tenendo a mente la tolleranza ai guasti è fondamentale per garantirne l'affidabilità e la robustezza in presenza di guasti permanenti. Ciò può comportare l'incorporazione di ridondanza, meccanismi di rilevamento e correzione degli errori e strategie di sicurezza nell'architettura del sistema. Affrontando in modo proattivo le sfide poste dai guasti permanenti, i sistemi ML possono mantenere la loro integrità, accuratezza e affidabilità, anche di fronte a guasti hardware.

17.3.3 Guasti Intermittenti

I guasti intermittenti sono guasti hardware che si verificano sporadicamente e in modo imprevedibile in un sistema. Un esempio è illustrato in Figura 17.9, dove le crepe nel materiale possono introdurre una maggiore resistenza [elettrica] nei circuiti. Questi guasti sono particolarmente difficili da rilevare e diagnosticare perché compaiono e scompaiono in modo intermittente, rendendo difficile riprodurre e isolare la causa principale. I guasti intermittenti possono causare instabilità del sistema, corruzione dei dati e degrado delle prestazioni.

Figura 17.9: Maggiore resistenza dovuta a un guasto intermittente, ovvero una crepa tra la protuberanza di rame e la saldatura del package. Fonte: Constantinescu

17.3.3.1 Definizione e Caratteristiche

I guasti intermittenti sono caratterizzati dalla loro natura sporadica e non deterministica. Si verificano in modo irregolare e possono apparire e scomparire spontaneamente, con durate e frequenze variabili. Questi guasti non si manifestano in modo coerente ogni volta che viene utilizzato il componente interessato, il che li rende più difficili da rilevare rispetto ai guasti permanenti. I guasti intermittenti possono interessare vari componenti hardware, tra cui processori, moduli di memoria, dispositivi di archiviazione o interconnessioni. Possono causare errori transitori, danneggiamento dei dati o comportamento imprevisto del sistema.
I guasti intermittenti possono avere un impatto significativo sul comportamento e l'affidabilità dei sistemi di elaborazione (Rashid, Pattabiraman, e Gopalakrishnan 2015). Ad esempio, un guasto intermittente nella logica di controllo di un processore può causare un flusso di programma irregolare, portando a calcoli errati o blocchi del sistema. I guasti intermittenti nei moduli di memoria possono danneggiare i valori dei dati, con conseguente esecuzione errata del programma o incoerenze nei dati. Nei dispositivi di archiviazione, i guasti intermittenti possono causare errori di lettura/scrittura o perdita di dati. Errori intermittenti nei canali di comunicazione possono causare corruzione dei dati, perdita di pacchetti o problemi di connettività intermittenti. Questi errori possono causare crash del sistema, problemi di integrità dei dati o degrado delle prestazioni, a seconda della gravità e della frequenza degli errori intermittenti.

17.3.3.2 Cause degli Errori Intermittenti

I guasti intermittenti possono derivare da diverse cause, sia interne che esterne, ai componenti hardware (Constantinescu 2008). Una causa comune è l'invecchiamento e l'usura dei componenti. Man mano che i dispositivi elettronici invecchiano, diventano più suscettibili a guasti intermittenti dovuti a meccanismi di degradazione come elettromigrazione, rottura dell'ossido o affaticamento dei giunti di saldatura.
Anche difetti di fabbricazione o variazioni di processo possono causare guasti intermittenti, in cui componenti marginali o borderline possono presentare guasti sporadici in condizioni specifiche, come mostrato in Figura 17.10.

Figura 17.10: Guasto intermittente indotto da residui in un chip DRAM. Fonte: Hynix Semiconductor

Fattori ambientali, come fluttuazioni di temperatura, umidità o vibrazioni, possono innescare guasti intermittenti alterando le caratteristiche elettriche dei componenti. Collegamenti allentati o degradati, come quelli nei connettori o nei circuiti stampati, possono causare guasti intermittenti.

17.3.3.3 Meccanismi dei Guasti Intermittenti

I guasti intermittenti possono manifestarsi attraverso vari meccanismi, a seconda della causa sottostante e del componente interessato. Un meccanismo è il circuito aperto o cortocircuito intermittente, in cui un percorso o una connessione del segnale viene temporaneamente interrotto o cortocircuitato, causando un comportamento irregolare. Un altro meccanismo è il guasto di ritardo intermittente (J. Zhang et al. 2018), in cui la temporizzazione dei segnali o i ritardi di propagazione diventano incoerenti, causando problemi di sincronizzazione o calcoli errati. I guasti intermittenti possono manifestarsi come bit flip [inversioni] transitori o errori soft nelle celle di memoria o nei registri, causando corruzione dei dati o esecuzione errata del programma.

17.3.3.4 Impatto sui Sistemi ML

Nel contesto dei sistemi ML, i guasti intermittenti possono introdurre sfide significative e avere un impatto sull'affidabilità e le prestazioni del sistema. Durante la fase di addestramento, i guasti intermittenti nelle unità di elaborazione o nella memoria possono portare a incongruenze nei calcoli, con conseguenti gradienti e aggiornamenti del peso errati o rumorosi. Ciò può influire sulla convergenza e l'accuratezza del processo di addestramento, portando a modelli sub-ottimali o instabili. Errori intermittenti di archiviazione o recupero dei dati possono corrompere i dati di training, introducendo rumore o errori che degradano la qualità dei modelli addestrati (Y. He et al. 2023).

Durante la fase di inferenza, gli errori intermittenti possono influire sull'affidabilità e la coerenza delle previsioni ML. Gli errori nelle unità di elaborazione o nella memoria possono causare calcoli errati o corruzione dei dati, portando a previsioni errate o incoerenti. Gli errori intermittenti nella pipeline dei dati possono introdurre rumore o errori nei dati di input, influenzando l'accuratezza e la robustezza delle previsioni. Nelle applicazioni safety-critical, come veicoli autonomi o sistemi di diagnosi medica, gli errori intermittenti possono avere gravi conseguenze, portando a decisioni o azioni errate che compromettono la sicurezza e l'affidabilità.

Per mitigare l'impatto degli errori intermittenti nei sistemi ML è necessario un approccio poliedrico (Rashid, Pattabiraman, e Gopalakrishnan 2012). A livello hardware, tecniche come pratiche di progettazione robuste, selezione dei componenti e controllo ambientale possono aiutare a ridurre il verificarsi di guasti intermittenti. Meccanismi di ridondanza e correzione degli errori possono essere impiegati per rilevare e ripristinare guasti intermittenti. A livello software, monitoraggio del runtime, rilevamento delle anomalie e tecniche di tolleranza ai guasti possono essere incorporate nella pipeline ML. Ciò può includere tecniche come convalida dei dati, rilevamento di valori anomali, assemblaggio di modelli o adattamento del modello di runtime per gestire con eleganza i guasti intermittenti.

Progettare sistemi ML resilienti ai guasti intermittenti è fondamentale per garantirne affidabilità e robustezza. Ciò comporta l'incorporazione di tecniche di tolleranza ai guasti, monitoraggio del runtime e meccanismi adattivi nell'architettura del sistema. Affrontando in modo proattivo le sfide dei guasti intermittenti, i sistemi ML possono mantenere la loro accuratezza, coerenza e affidabilità, anche in caso di guasti hardware sporadici. Test, monitoraggio e manutenzione regolari dei sistemi ML possono aiutare a identificare e mitigare i guasti intermittenti prima che causino interruzioni significative o un degrado delle prestazioni.

17.3.4 Rilevamento e Mitigazione

Questa sezione esplora varie tecniche di rilevamento degli errori, inclusi approcci a livello hardware e software, e discute strategie di mitigazione efficaci per migliorare la resilienza dei sistemi ML. Inoltre, esamineremo le considerazioni sulla progettazione di sistemi ML

resilienti, presenteremo casi di studio ed esempi e metteremo in evidenza le future direzioni di ricerca nei sistemi ML tolleranti agli errori.

17.3.4.1 Tecniche di Rilevamento degli Errori

Le tecniche di rilevamento degli errori sono importanti per identificare e localizzare gli errori hardware nei sistemi ML. Queste tecniche possono essere ampiamente categorizzate in approcci a livello hardware e software, ognuno dei quali offre capacità e vantaggi unici.

17.3.4.1.1 Rilevamento degli errori a livello hardware

Le tecniche di rilevamento degli errori a livello hardware sono implementate a livello fisico del sistema e mirano a identificare gli errori nei componenti hardware sottostanti. Esistono diverse tecniche hardware, ma in generale, possiamo raggruppare questi diversi meccanismi nelle seguenti categorie.

Built-in self-test (BIST) mechanisms: BIST è una tecnica potente per rilevare guasti nei componenti hardware (Bushnell e Agrawal 2002). Comporta l'incorporazione di circuiti hardware aggiuntivi nel sistema per l'autotest e il rilevamento dei guasti. BIST può essere applicato a vari componenti, come processori, moduli di memoria o circuiti integrati specifici per applicazione (ASIC). Ad esempio, BIST può essere implementato in un processore utilizzando catene di scansione, che sono percorsi dedicati che consentono l'accesso ai registri interni e alla logica per scopi di test.
Durante il processo BIST, vengono applicati pattern di test predefiniti ai circuiti interni del processore e le risposte vengono confrontate con i valori previsti. Eventuali discrepanze indicano la presenza di guasti. I processori Xeon di Intel, ad esempio, includono meccanismi BIST per testare i core della CPU, la memoria cache e altri componenti critici durante l'avvio del sistema.

Codici di rilevamento degli errori: I codici di rilevamento degli errori sono ampiamente utilizzati per rilevare errori di archiviazione e trasmissione dei dati (Hamming 1950). Questi codici aggiungono bit ridondanti ai dati originali, consentendo il rilevamento di errori di bit. Esempio: I controlli di parità sono una forma semplice di codice di rilevamento degli errori mostrato in Figura 17.11. In uno schema di parità a bit singolo, un bit extra viene aggiunto a ogni parola di dati, rendendo il numero di 1 nella parola pari (parità pari) o dispari (parità dispari).

Figura 17.11: Esempio di bit di parità. Fonte: Computer Hope

Quando si leggono i dati, la parità viene controllata e, se non corrisponde al valore previsto, viene rilevato un errore. Codici di rilevamento degli errori più avanzati, come i "cyclic redundancy checks (CRC)" [controlli di ridondanza ciclica], calcolano un checksum in base ai dati e lo aggiungono al messaggio. Il checksum viene ricalcolato all'estremità ricevente e confrontato con il checksum trasmesso per rilevare gli errori. I moduli di memoria con "Error-correcting code (ECC)" [codice di correzione degli errori], comunemente utilizzati nei server e

nei sistemi critici, impiegano codici avanzati di rilevamento e correzione degli errori per rilevare e correggere errori a bit singolo o multi-bit nella memoria.

Ridondanza hardware e meccanismi di voto: La ridondanza hardware implica la duplicazione dei componenti critici e il confronto dei loro output per rilevare e mascherare i guasti (Sheaffer, Luebke, e Skadron 2007). I meccanismi di voto, come la "triple modular redundancy (TMR)" [ridondanza modulare tripla], impiegano più istanze di un componente e confrontano i loro output per identificare e mascherare comportamenti difettosi (Arifeen, Hassan, e Lee 2020).
In un sistema TMR, tre istanze identiche di un componente hardware, come un processore o un sensore, eseguono lo stesso calcolo in parallelo. Gli output di queste istanze vengono immessi in un circuito di voto, che confronta i risultati e seleziona il valore di maggioranza come output finale. Se una delle istanze produce un risultato non corretto a causa di un guasto, il meccanismo di voto maschera l'errore e mantiene l'output corretto. Il TMR è comunemente utilizzato nei sistemi aerospaziali e aeronautici, dove l'elevata affidabilità è fondamentale. Ad esempio, l'aereo Boeing 777 impiega il TMR nel suo sistema di computer di volo primario per garantire la disponibilità e la correttezza delle funzioni di controllo del volo (Yeh 1996).
I computer a guida autonoma di Tesla impiegano un'architettura hardware ridondante per garantire la sicurezza e l'affidabilità delle funzioni critiche, come percezione, processo decisionale e controllo del veicolo, come mostrato in Figura 17.12. Un componente chiave di questa architettura è l'utilizzo della "dual modular redundancy (DMR)" [ridondanza modulare duale] nei sistemi di computer di bordo dell'auto.

Figura 17.12: Computer Tesla a guida autonoma completa con SoC duali ridondanti. Fonte: Tesla

Nell'implementazione DMR di Tesla, due unità hardware identiche, spesso chiamate "computer ridondanti" o "unità di controllo ridondanti", eseguono gli stessi calcoli in parallelo (Bannon et al. 2019). Ogni unità elabora in modo indipendente i dati dei sensori, esegue algoritmi di percezione e decisionali e genera comandi di controllo per gli attuatori del veicolo (ad esempio, sterzo, accelerazione e frenata).
Gli output di queste due unità ridondanti vengono costantemente confrontati per rilevare eventuali discrepanze o guasti. Se gli output corrispondono, il sistema presuppone che entrambe le unità funzionino correttamente e i comandi di controllo vengono inviati agli attuatori del veicolo. Tuttavia, se c'è una mancata corrispondenza tra gli output, il sistema identifica un potenziale guasto in una delle unità e adotta le misure appropriate per garantire un funzionamento sicuro.
Il sistema può impiegare meccanismi aggiuntivi per determinare quale unità è difettosa in una mancata corrispondenza. Ciò può comportare l'utilizzo di algoritmi diagnostici, il confronto degli output con i dati di altri sensori o sottosistemi o l'analisi della coerenza degli

output nel tempo. Una volta identificata l'unità difettosa, il sistema può isolarla e continuare a funzionare utilizzando l'output dell'unità non difettosa.
Il DMR nel computer di guida autonoma di Tesla fornisce un ulteriore livello di sicurezza e tolleranza ai guasti. Avendo due unità indipendenti che eseguono gli stessi calcoli, il sistema può rilevare e mitigare i guasti che possono verificarsi in una delle unità. Questa ridondanza aiuta a prevenire singoli punti di guasto e garantisce che le funzioni critiche rimangano operative nonostante i guasti hardware.
Inoltre, Tesla incorpora anche meccanismi di ridondanza aggiuntivi oltre al DMR. Ad esempio, utilizzano alimentatori ridondanti, sistemi di sterzo e frenata e diverse suite di sensori (ad esempio, telecamere, radar e sensori a ultrasuoni) per fornire più livelli di tolleranza ai guasti. Queste ridondanze contribuiscono collettivamente alla sicurezza e all'affidabilità complessive del sistema di guida autonoma.
È importante notare che mentre DMR fornisce rilevamento guasti e un certo livello di tolleranza ai guasti, TMR può fornire un diverso livello di mascheramento dei guasti. In DMR, se entrambe le unità subiscono guasti simultanei o il guasto influisce sul meccanismo di confronto, il sistema potrebbe non essere in grado di identificare il guasto. Pertanto, gli SDC di Tesla si basano su una combinazione di DMR e altri meccanismi di ridondanza per raggiungere un elevato livello di tolleranza ai guasti.
L'uso di DMR nel computer a guida autonoma di Tesla evidenzia l'importanza della ridondanza hardware nelle applicazioni critiche per la sicurezza. Utilizzando unità di elaborazione ridondanti e confrontando i loro output, il sistema può rilevare e mitigare i guasti, migliorando la sicurezza e l'affidabilità complessive della funzionalità di guida autonoma.
Google utilizza "hot spare" ridondanti per gestire i problemi SDC nei suoi data center, migliorando così l'affidabilità delle funzioni critiche. Come illustrato in Figura 17.13, durante la normale fase di addestramento, più "worker" di training sincroni funzionano in modo impeccabile. Tuttavia, se un worker diventa difettoso e causa SDC, un verificatore SDC identifica automaticamente i problemi. Dopo aver rilevato l'SDC, il verificatore SDC sposta il training su un hot spare e invia la macchina difettosa per la riparazione. Questa ridondanza salvaguarda la continuità e l'affidabilità del training ML, riducendo al minimo i tempi di inattività e preservando l'integrità dei dati.

Figura 17.13: Google impiega "core hot spare" per gestire in modo trasparente gli SDC nel data center. Fonte: Jeff Dean, MLSys 2024 Keynote (Google)

Watchdog timer: I watchdog timer sono componenti hardware che monitorano l'esecuzione di attività o processi critici (Pont e Ong 2002). Sono comunemente utilizzati per rilevare e ripristinare guasti software o hardware che causano la mancata risposta di un sistema o il suo blocco in un ciclo infinito. In un sistema embedded, un watchdog timer può essere configurato per monitorare l'esecuzione del loop principale, come illustrato in Figura 17.14. Il software reimposta periodicamente il watchdog timer per indicare che funziona correttamente. Supponiamo che il software non riesca a reimpostare il timer entro un limite di tempo specificato (periodo di timeout). In tal caso, il watchdog timer presuppone che il sistema abbia riscontrato un guasto e attiva un'azione di ripristino predefinita, come il reset del sistema o il passaggio a un componente di backup. I watchdog timer sono ampiamente utilizzati nell'elettronica automobilistica, nei sistemi di controllo industriale e in altre applicazioni critiche per la sicurezza per garantire il rilevamento e il ripristino tempestivi dai guasti.

Figura 17.14: Esempio di watchdog timer nel rilevamento di guasti MCU. Fonte: Ablic

17.3.4.1.2 Rilevamento guasti a livello software

Le tecniche di rilevamento degli errori a livello software si basano su algoritmi software e meccanismi di monitoraggio per identificare gli errori di sistema. Queste tecniche possono essere implementate a vari livelli dello stack software, tra cui il sistema operativo, il middleware o il livello dell'applicazione.

Monitoraggio del runtime e rilevamento delle anomalie: Il monitoraggio del runtime comporta l'osservazione continua del comportamento del sistema e dei suoi componenti durante l'esecuzione (Francalanza et al. 2017). Aiuta a rilevare anomalie, errori o comportamenti imprevisti che potrebbero indicare la presenza di errori. Ad esempio, si consideri un sistema di classificazione delle immagini basato su ML distribuito in un'auto a guida autonoma. Il monitoraggio del runtime può essere implementato per tracciare le prestazioni e il comportamento del modello di classificazione (Mahmoud et al. 2021).
Gli algoritmi di rilevamento delle anomalie possono essere applicati alle previsioni del modello o alle attivazioni di livelli intermedi, come il rilevamento statistico di valori anomali o approcci basati sull'apprendimento automatico (ad esempio, One-Class SVM o Autoencoders) (Chandola, Banerjee, e Kumar 2009). Figura 17.15 mostra un esempio di rilevamento delle anomalie. Supponiamo che il sistema di monitoraggio rilevi una deviazione significativa dai pattern previsti, come un calo improvviso dell'accuratezza della classificazione o campioni fuori distribuzione. In tal caso, può generare un "alert" che indica un potenziale errore nel modello o nella pipeline dei dati di input. Questo rilevamento precoce consente di applicare strategie di intervento tempestivo e di mitigazione degli errori.

Figura 17.15: Esempi di rilevamento delle anomalie. (a) Rilevamento delle anomalie completamente supervisionato, (b) rilevamento delle anomalie solo normali, (c, d, e) rilevamento delle anomalie semi-supervisionato, (f) rilevamento delle anomalie non supervisionato. Fonte: Google

Controlli di coerenza e convalida dei dati: I controlli di coerenza e le tecniche di convalida dei dati garantiscono l'integrità e la correttezza dei dati in diverse fasi di elaborazione in un sistema ML (A. Lindholm et al. 2019). Questi controlli aiutano a rilevare danneggiamenti dei dati, incongruenze o errori che potrebbero propagarsi e influenzare il comportamento del sistema. Esempio: In un sistema ML distribuito in cui più nodi collaborano per addestrare un modello, è possibile implementare controlli di coerenza per convalidare l'integrità dei parametri condivisi del modello. Ogni nodo può calcolare un checksum o un hash dei parametri del modello prima e dopo l'iterazione di addestramento, come mostrato in Figura 17.15. Eventuali incongruenze o danneggiamenti dei dati possono essere rilevati confrontando i checksum tra i nodi. Inoltre, è possibile applicare controlli di intervallo ai dati di input e agli output del modello per garantire che rientrino nei limiti previsti. Ad esempio, se il sistema di percezione di un veicolo autonomo rileva un oggetto con dimensioni o velocità non realistiche, può indicare un errore nei dati del sensore o negli algoritmi di percezione (Wan et al. 2023).

Meccanismi di heartbeat e timeout: I meccanismi di heartbeat e timeout sono comunemente utilizzati per rilevare errori nei sistemi distribuiti e garantire la vitalità e la reattività dei componenti (Kawazoe Aguilera, Chen, e Toueg 1997). Sono molto simili ai timer watchdog presenti nell'hardware. Ad esempio, in un sistema ML distribuito, in cui più nodi collaborano per eseguire attività quali pre-elaborazione dei dati, training del modello o inferenza, è possibile implementare meccanismi heartbeat per monitorare lo stato e la disponibilità di ciascun nodo. Ogni nodo invia periodicamente un messaggio heartbeat a un coordinatore centrale o ai suoi nodi peer, indicando il suo stato e la sua disponibilità. Supponiamo che un nodo non riesca a inviare un heartbeat entro un periodo di timeout

specificato, come mostrato in Figura 17.16. In tal caso, viene considerato difettoso e possono essere intraprese azioni appropriate, come la ridistribuzione del carico di lavoro o l'avvio di un meccanismo di "failover". I timeout possono anche essere utilizzati per rilevare e gestire componenti bloccati o non reattivi. Ad esempio, se un processo di caricamento dati supera una soglia di timeout predefinita, potrebbe indicare un errore nella pipeline dati e il sistema può adottare misure correttive.

Figura 17.16: Messaggi heartbeat nei sistemi distribuiti. Fonte: GeeksforGeeks

Tecniche di "Software-implemented fault tolerance (SIFT)": Le tecniche SIFT introducono meccanismi di ridondanza e rilevamento degli errori a livello software per migliorare l'affidabilità e la tolleranza agli errori del sistema (Reis et al. 2005). Esempio: La programmazione N-version è una tecnica SIFT in cui più versioni di componenti software funzionalmente equivalenti vengono sviluppate in modo indipendente da team diversi. Questo può essere applicato a componenti critici come il motore di inferenza del modello in un sistema ML. Più versioni del motore di inferenza possono essere eseguite in parallelo e i loro output possono essere confrontati per coerenza. È considerato il risultato corretto se la maggior parte delle versioni produce lo stesso output. Se c'è una discrepanza, indica un potenziale errore in una o più versioni e possono essere attivati meccanismi di gestione degli errori appropriati. Un altro esempio è l'utilizzo di codici di correzione degli errori basati su software, come i codici Reed-Solomon (Plank 1997), per rilevare e correggere errori nell'archiviazione o nella trasmissione dei dati, come mostrato in Figura 17.17. Questi codici aggiungono ridondanza ai dati, consentendo di rilevare e correggere determinati errori e migliorare la tolleranza agli errori del sistema.

Figura 17.17: Rappresentazione a n bit dei codici Reed-Solomon. Fonte: GeeksforGeeks

Esercizio 17.1: Rilevamento delle Anomalie

In questo Colab, si svolge il ruolo di un detective di guasti IA! Si costruirà un rilevatore di anomalie basato su autoencoder per individuare gli errori nei dati sulla salute cardiaca. Si scopre come identificare i malfunzionamenti nei sistemi ML, un'abilità fondamentale per creare un'IA affidabile. Utilizzeremo Keras Tuner per mettere a punto l'autoencoder per un rilevamento di guasti di prim'ordine. Questa esperienza si collega direttamente al capitolo Robust AI, dimostrando l'importanza del rilevamento di guasti in applicazioni reali come l'assistenza sanitaria e i sistemi autonomi. Preparatevi a rafforzare l'affidabilità delle creazioni IA!

Open in Colab

17.3.5 Riepilogo

Tabella 17.1 fornisce un'analisi comparativa estesa di guasti transitori, permanenti e intermittenti. Descrive le caratteristiche o dimensioni primarie che distinguono questi tipi di guasti. Qui, riassumiamo le dimensioni rilevanti che abbiamo esaminato ed esploriamo le sfumature che differenziano i guasti transitori, permanenti e intermittenti in modo più dettagliato.

Tabella 17.1: Confronto tra guasti transitori, permanenti e intermittenti.

Dimensione	Guasti Transitori	Guasti Permanenti	Guasti intermittenti
Durata	Di breve durata, temporaneo	Persistente, rimane fino alla riparazione o alla sostituzione	Sporadica, appare e scompare in modo intermittente
Persistenza	Scompare dopo che la condizione di errore è passata	È costantemente presente finché non viene affrontato	Si ripete in modo irregolare, non sempre presente
Cause	Fattori esterni (ad esempio, interferenza elettromagnetica raggi cosmici)	Difetti hardware, danni fisici, usura	Condizioni hardware instabili, connessioni allentate, componenti obsoleti
Manifestazione	Bit flip, glitch, danneggiamento temporaneo dei dati	Errori bloccati, componenti rotti, guasti completi del dispositivo	Bit flip occasionali, problemi di segnale intermittenti, malfunzionamenti sporadici
Impatto sui Sistemi ML	Introduce errori temporanei o rumore nei calcoli	Causa errori o guasti costanti, che influiscono sull'affidabilità	Porta a errori sporadici e imprevedibili, difficili da diagnosticare e mitigare
Rilevamento	Codici di rilevamento degli errori, confronto con i valori previsti	Autotest integrati, codici di rilevamento degli errori, controlli di coerenza	Monitoraggio delle anomalie, analisi di pattern di errore e correlazioni
Mitigazione	Codici di correzione degli errori, ridondanza, checkpoint e riavvio	Riparazione o sostituzione hardware, ridondanza dei componenti, meccanismi di failover	Progettazione robusta, controllo ambientale, monitoraggio del runtime, tecniche di tolleranza agli errori

17.4 Robustezza del Modello ML

17.4.1 Attacchi Avversari

17.4.1.1 Definizione e Caratteristiche

Gli attacchi avversari mirano a indurre i modelli a fare previsioni errate fornendo loro input ingannevoli appositamente creati (chiamati esempi avversari) (Parrish et al. 2023).

Aggiungendo lievi perturbazioni ai dati di input, gli avversari possono "hackerare" il riconoscimento di pattern di un modello e ingannarlo. Si tratta di tecniche sofisticate in cui piccole, spesso impercettibili modifiche ai dati di input possono indurre un modello ML a fare una previsione errata, come mostrato in Figura 17.18.

Figura 17.18: Un piccolo rumore avversario aggiunto all'immagine originale può far sì che la rete neurale classifichi l'immagine come un Guacamole anziché come un gatto egiziano. Fonte: Sutanto

È possibile generare prompt che portano a immagini non sicure in modelli testo-immagine come DALLE (Ramesh et al. 2021) o Stable Diffusion (Rombach et al. 2022). Ad esempio, alterando i valori dei pixel di un'immagine, gli aggressori possono ingannare un sistema di riconoscimento facciale facendogli identificare un volto come una persona diversa.
Gli attacchi avversari sfruttano il modo in cui i modelli ML apprendono e prendono decisioni durante l'inferenza. Questi modelli funzionano sul principio di riconoscimento di pattern nei dati. Un avversario crea input speciali con perturbazioni per confondere il riconoscimento degli pattern del modello, in pratica "hackerando" le percezioni del modello.
Gli attacchi avversari rientrano in diversi scenari:

- **Attacchi Whitebox:** L'attaccante conosce perfettamente il funzionamento interno del modello target, inclusi i dati di training, i parametri e l'architettura (Ye e Hamidi 2021). Questo accesso completo crea condizioni favorevoli per gli aggressori per sfruttare le vulnerabilità del modello. L'attaccante può usare debolezze specifiche e sottili per creare esempi avversari efficaci.
- **Attacchi Blackbox:** A differenza degli attacchi White-box, i Black-box implicano che l'attaccante abbia poca o nessuna conoscenza del modello target (C. Guo et al. 2019). Per eseguire l'attacco, l'attore avversario deve osservare attentamente il comportamento dell'output del modello.
- **Attacchi Greybox:** Si collocano tra gli attacchi Blackbox e Whitebox. L'attaccante ha solo una conoscenza parziale della progettazione interna del modello target (Y. Xu et al. 2021). Ad esempio, l'attaccante potrebbe avere conoscenza dei dati di training ma non dell'architettura o dei parametri. Nel mondo reale, gli attacchi pratici rientrano solitamente nelle categorie black-box o grey-box.

Il panorama dei modelli di apprendimento automatico è complesso e ampio, soprattutto data la loro relativamente recente integrazione nelle applicazioni commerciali. Questa rapida adozione, sebbene trasformativa, ha portato alla luce numerose vulnerabilità all'interno di questi modelli. Di conseguenza, sono emersi vari metodi di attacco avversari, ognuno dei quali sfrutta strategicamente diversi aspetti di vari modelli. Di seguito, evidenziamo un sottoinsieme di questi metodi, che mostra la natura multiforme degli attacchi avversari sui modelli di apprendimento automatico:

- Le **Generative Adversarial Network (GAN)** sono modelli di deep learning costituiti da due reti in competizione tra loro: un generatore e un discriminatore (Goodfellow et al. 2020). Il generatore cerca di sintetizzare dati realistici mentre il discriminatore valuta se sono reali o falsi. Le GAN possono essere utilizzate per creare esempi avversari. La rete del generatore è addestrata per produrre input che il modello target classifica erroneamente. Queste immagini generate da GAN possono quindi attaccare un classificatore target o un modello di rilevamento. Il generatore e il modello target sono impegnati in un processo competitivo, con il generatore che migliora continuamente la sua capacità di creare esempi ingannevoli e il modello target che aumenta la sua resistenza a tali esempi. Le reti GAN forniscono un potente framework per la creazione di input avversari complessi e diversificati, dimostrando l'adattabilità dei modelli generativi nel panorama avversario.
- I **Transfer Learning Adversarial Attacks** [attacchi avversari di apprendimento di trasferimento] sfruttano la conoscenza trasferita da un modello pre-addestrato a un modello target, creando esempi avversari che possono ingannare entrambi i modelli. Questi attacchi rappresentano una preoccupazione crescente, in particolare quando gli avversari hanno conoscenza dell'estrattore di feature ma non hanno accesso alla testa di classificazione (la parte o il layer responsabile della creazione delle classificazioni finali). Denominate "attacchi headless", queste strategie avversarie trasferibili sfruttano le capacità espressive degli estrattori di feature per creare perturbazioni, senza tenere conto dello spazio delle etichette o dei dati di addestramento. L'esistenza di tali attacchi sottolinea l'importanza di sviluppare difese robuste per le applicazioni di apprendimento tramite trasferimento, soprattutto perché i modelli pre-addestrati sono comunemente utilizzati (Abdelkader et al. 2020).

17.4.1.2 Meccanismi degli Attacchi Avversari

Attacchi Basati sul Gradiente

Una categoria importante di attacchi avversari è quella degli attacchi basati sul gradiente. Questi attacchi sfruttano i gradienti della funzione di perdita del modello ML per creare esempi avversari. Il Fast Gradient Sign Method (FGSM) è una tecnica ben nota in questa categoria. FGSM perturba i dati di input aggiungendo un piccolo rumore nella direzione del gradiente, con l'obiettivo di massimizzare l'errore di previsione del modello. FGSM può generare rapidamente esempi avversari, come mostrato in Figura 17.19, eseguendo un singolo passaggio nella direzione del gradiente.

Figura 17.19: Attacchi Basati sul Gradiente. Fonte: Ivezic

Un'altra variante, l'attacco "Projected Gradient Descent (PGD)", estende FGSM applicando iterativamente la fase di aggiornamento del gradiente, consentendo esempi avversari più raffinati e potenti. L'attacco "Jacobian-based Saliency Map (JSMA)" è un altro approccio basato sul gradiente che identifica le caratteristiche di input più influenti e le perturba per creare esempi avversari.

Attacchi Basati sull'Ottimizzazione
Questi attacchi formulano la generazione di esempi avversari come un problema di ottimizzazione. L'attacco Carlini e Wagner (C&W) è un esempio importante in questa categoria. Trova la perturbazione più piccola che può causare una classificazione errata mantenendo la somiglianza percettiva con l'input originale. L'attacco C&W impiega un processo di ottimizzazione iterativo per ridurre al minimo la perturbazione massimizzando al contempo l'errore di previsione del modello.
Un altro approccio basato sull'ottimizzazione è l'Elastic Net Attack to DNNs (EAD), che incorpora la regolarizzazione elastica della rete per generare esempi avversari con perturbazioni sparse.

Attacchi Basati sul Trasferimento
Gli attacchi basati sul trasferimento sfruttano la proprietà di trasferibilità degli esempi avversari. La trasferibilità si riferisce al fenomeno per cui gli esempi avversari creati per un modello ML possono spesso ingannare altri modelli, anche se hanno architetture diverse o sono stati addestrati su set di dati diversi. Ciò consente agli aggressori di generare esempi avversari utilizzando un modello surrogato e quindi trasferirli al modello target senza richiedere l'accesso diretto ai suoi parametri o gradienti. Gli attacchi basati sul trasferimento evidenziano la generalizzazione delle vulnerabilità avversarie su diversi modelli e il potenziale per attacchi black-box.

Attacchi nel Mondo Fisico
Gli attacchi nel mondo fisico portano gli esempi avversari nel regno degli scenari del mondo reale. Questi attacchi comportano la creazione di oggetti fisici o manipolazioni che possono ingannare i modelli ML quando vengono catturati da sensori o telecamere. Le patch avversarie, ad esempio, sono piccole patch progettate con cura che possono essere posizionate sugli oggetti per ingannare i modelli di rilevamento o classificazione degli oggetti. Quando vengono applicate a oggetti del mondo reale, queste patch possono causare una classificazione errata dei modelli o il mancato rilevamento accurato degli oggetti. Gli oggetti avversari, come sculture stampate in 3D o segnali stradali modificati, possono anche essere creati per ingannare i sistemi ML in ambienti fisici.

Riepilogo
Tabella 17.2 una panoramica concisa delle diverse categorie di attacchi avversari, tra cui attacchi basati su gradiente (FGSM, PGD, JSMA), attacchi basati sull'ottimizzazione (C&W, EAD), attacchi basati sul trasferimento e attacchi nel mondo fisico (patch e oggetti avversari). Ogni attacco viene brevemente descritto, evidenziandone le caratteristiche e i meccanismi principali.

Tabella 17.2: Diversi tipi di attacco sui modelli ML.

Categoria di attacco	Nome attacco	Descrizione
Basato sul gradiente	Fast Gradient Sign Method (FGSM) Projected Gradient Descent (PGD) Jacobian-based Saliency Map Attack (JSMA)	Perturba i dati di input aggiungendo un piccolo rumore nella direzione del gradiente per massimizzare l'errore di previsione. Estende FGSM applicando iterativamente il passaggio di aggiornamento del gradiente per esempi avversari più raffinati. Identifica le caratteristiche di input influenti e le perturba per creare esempi avversari.
Basato sull'ottimizzazione	Carlini and Wagner (C&W) Attack Elastic Net Attack to DNNs (EAD)	Trova la perturbazione più piccola che causa una classificazione errata mantenendo la somiglianza percettiva. Incorpora la regolarizzazione elastica della

		rete per generare esempi avversari con perturbazioni sparse.
Basato sul trasferimento	Transferability-based Attacks	Sfrutta la trasferibilità di esempi avversari su modelli diversi, consentendo attacchi black-box.
Mondo fisico	Adversarial Patches Adversarial Objects	Piccole patch attentamente progettate, posizionate sugli oggetti per ingannare i modelli di rilevamento o classificazione degli oggetti. Oggetti fisici (ad esempio, sculture stampate in 3D, segnali stradali modificati) creati per ingannare i sistemi ML in scenari del mondo reale.

I meccanismi degli attacchi avversari rivelano l'intricata interazione tra i limiti decisionali del modello ML, i dati di input e gli obiettivi dell'attaccante. Manipolando attentamente i dati di input, gli aggressori possono sfruttare le sensibilità e i punti ciechi del modello, portando a previsioni errate. Il successo degli attacchi avversari evidenzia la necessità di una comprensione più approfondita delle proprietà di robustezza e generalizzazione dei modelli ML.

La difesa dagli attacchi avversari richiede un approccio multiforme. L'addestramento avversario è una strategia di difesa comune in cui i modelli vengono addestrati su esempi avversari per migliorare la robustezza. Esporre il modello a esempi avversari durante l'addestramento gli insegna a classificarli correttamente e a diventare più resiliente agli attacchi. La distillazione difensiva, la preelaborazione degli input e i metodi di ensemble sono altre tecniche che possono aiutare a mitigare l'impatto degli attacchi avversari.

Man mano che l'apprendimento automatico avversario si evolve, i ricercatori esplorano nuovi meccanismi di attacco e sviluppano difese più sofisticate. La corsa agli armamenti tra aggressori e difensori spinge la necessità di innovazione e vigilanza costanti nel proteggere i sistemi ML dalle minacce avversarie. Comprendere i meccanismi degli attacchi avversari è fondamentale per sviluppare modelli ML robusti e affidabili in grado di resistere al panorama in continua evoluzione degli esempi avversari.

17.4.1.3 Impatto sui Sistemi ML

Gli attacchi avversari sui sistemi di apprendimento automatico sono emersi come una preoccupazione significativa negli ultimi anni, evidenziando le potenziali vulnerabilità e i rischi associati all'adozione diffusa delle tecnologie ML. Questi attacchi comportano perturbazioni attentamente studiate per immettere dati che possono ingannare o fuorviare i modelli ML, portando a previsioni errate o classificazioni errate, come mostrato in Figura 17.20. L'impatto degli attacchi avversari sui sistemi ML è di vasta portata e può avere gravi conseguenze in vari domini.

Figura 17.20: Generazione di esempi avversari applicata a GoogLeNet (Szegedy et al., 2014a) su ImageNet. Fonte: Goodfellow

Un esempio lampante dell'impatto degli attacchi avversari è stato dimostrato dai ricercatori nel 2017. Hanno sperimentato piccoli adesivi in bianco e nero sui segnali di stop (Eykholt et al. 2017). All'occhio umano, questi adesivi non oscuravano il segnale né ne impedivano l'interpretazione. Tuttavia, quando le immagini dei segnali di stop modificati dagli adesivi sono state inserite nei modelli ML standard di classificazione dei segnali stradali, è emerso un risultato scioccante. I modelli hanno classificato erroneamente i segnali di stop come segnali di limite di velocità nell'85% dei casi.
Questa dimostrazione ha fatto luce sul potenziale allarmante di semplici adesivi avversari per ingannare i sistemi ML e fargli interpretare male i segnali stradali critici. Le implicazioni di tali attacchi nel mondo reale sono significative, in particolare nel contesto dei veicoli autonomi. Se utilizzati su strade reali, questi adesivi avversari potrebbero far sì che le auto a guida autonoma interpretino erroneamente i segnali di stop come limiti di velocità, portando a situazioni pericolose, come mostrato in Figura 17.21. I ricercatori hanno avvertito che ciò potrebbe causare arresti a rotazione o accelerazioni involontarie negli incroci, mettendo a repentaglio la sicurezza pubblica.

Figura 17.21: I graffiti su un segnale di stop hanno ingannato un'auto a guida autonoma facendole credere che si trattasse di un segnale di limite di velocità di 45 mph. Fonte: Eykholt

Il caso di studio degli adesivi avversari sui segnali di stop fornisce un'illustrazione concreta di come gli esempi avversari sfruttino il modo in cui i modelli ML riconoscono i pattern. Manipolando in modo sottile i dati di input in modi invisibili agli esseri umani, gli aggressori possono indurre previsioni errate e creare gravi rischi, specialmente in applicazioni critiche per la sicurezza come i veicoli autonomi. La semplicità dell'attacco evidenzia la vulnerabilità dei modelli ML anche a piccole modifiche nell'input, sottolineando la necessità di difese robuste contro tali minacce.
L'impatto degli attacchi avversari si estende oltre il degrado delle prestazioni del modello. Questi attacchi sollevano notevoli preoccupazioni in termini di sicurezza e protezione, in particolare nei domini in cui i modelli ML sono utilizzati per prendere decisioni critiche. Nelle applicazioni sanitarie, gli attacchi avversari sui modelli di imaging medico potrebbero portare a diagnosi errate o raccomandazioni di trattamento errate, mettendo a repentaglio il benessere del paziente (M.-J. Tsai, Lin, e Lee 2023). Nei sistemi finanziari, gli attacchi avversari potrebbero consentire frodi o manipolazioni di algoritmi di trading, con conseguenti perdite economiche sostanziali.
Inoltre, le vulnerabilità avversarie compromettono l'affidabilità e l'interpretabilità dei modelli ML. Se perturbazioni attentamente realizzate possono facilmente ingannare i modelli, la fiducia nelle loro previsioni e decisioni si erode. Gli esempi avversari espongono la dipendenza dei modelli da pattern superficiali e l'incapacità di catturare i veri concetti sottostanti, mettendo in discussione l'affidabilità dei sistemi ML (Fursov et al. 2021).

La difesa dagli attacchi avversari richiede spesso risorse computazionali aggiuntive e può influire sulle prestazioni complessive del sistema. Tecniche come l'addestramento avversariale, in cui i modelli vengono addestrati su esempi avversari per migliorare la robustezza, possono aumentare significativamente i tempi di addestramento e i requisiti computazionali (Bai et al. 2021). I meccanismi di rilevamento e mitigazione del runtime, come la preelaborazione dell'input (Addepalli et al. 2020) o i controlli di coerenza delle previsioni, introducono latenza e influenzano le prestazioni in tempo reale dei sistemi ML.
La presenza di vulnerabilità avversarie complica anche l'implementazione e la manutenzione dei sistemi ML. I progettisti e gli operatori di sistema devono considerare il potenziale di attacchi avversari e incorporare difese e meccanismi di monitoraggio appropriati. Aggiornamenti regolari e riqualificazione dei modelli diventano necessari per adattarsi alle nuove tecniche avversarie e mantenere la sicurezza e le prestazioni del sistema nel tempo.
L'impatto degli attacchi avversari sui sistemi ML è significativo e multiforme. Questi attacchi espongono le vulnerabilità dei modelli ML, dal degrado delle prestazioni del modello e dall'aumento di preoccupazioni sulla sicurezza e la protezione alla sfida dell'affidabilità e dell'interpretabilità del modello. Sviluppatori e ricercatori devono dare priorità allo sviluppo di difese e contromisure robuste per mitigare i rischi posti dagli attacchi avversari. Affrontando queste sfide, possiamo creare sistemi ML più sicuri, affidabili e degni di fiducia in grado di resistere al panorama in continua evoluzione delle minacce avversarie.

Esercizio 17.2: Attacchi Avversari

Preparatevi a diventare un avversario dell'IA! In questo Colab, si diventerà un hacker white-box, imparando a creare attacchi che ingannano i modelli di classificazione delle immagini. Ci concentreremo sul Fast Gradient Sign Method (FGSM), sfruttando i gradienti di un modello contro di esso! Si distorceranno deliberatamente le immagini con piccole perturbazioni, osservando come inganneranno sempre più intensamente l'IA. Questo esercizio pratico evidenzia l'importanza di creare un'IA sicura, un'abilità critica man mano che l'IA si integra nelle auto e nell'assistenza sanitaria. Il Colab si collega direttamente al capitolo Robust AI del libro, spostando gli attacchi avversari dalla teoria alla esperienza pratica.

Open in Colab

Pensate di poter superare in astuzia un'IA? In questo Colab, scopriremo come ingannare i modelli di classificazione delle immagini con attacchi avversari. Utilizzeremo metodi come FGSM per modificare le immagini e ingannare sottilmente l'IA. Scopriremo come progettare patch di immagini ingannevoli e osserveremo la sorprendente vulnerabilità di questi potenti modelli. Questa è una conoscenza fondamentale per costruire sistemi di IA veramente robusti!

Open in Colab

17.4.2 Avvelenamento dei Dati

17.4.2.1 Definizione e Caratteristiche

L'avvelenamento dei dati è un attacco in cui i dati di addestramento vengono manomessi, portando alla compromissione del modello (Biggio, Nelson, e Laskov 2012), come mostrato in Figura 17.22. Gli aggressori possono modificare gli esempi di training esistenti, inserire nuovi dati dannosi o influenzare il processo di raccolta dati. I dati avvelenati vengono etichettati in modo tale da alterare il comportamento appreso del modello. Ciò può essere particolarmente dannoso nelle applicazioni in cui i modelli ML prendono decisioni automatizzate in base a pattern appresi. Oltre ai set di training, i test di avvelenamento e i dati

di convalida possono consentire agli avversari di aumentare artificialmente le prestazioni del modello segnalate.

Figura 17.22: Effetti dell'Avvelenamento di NightShade sulla Diffusione Stabile. Fonte: TOMÉ

Il processo di solito prevede i seguenti passaggi:

- **Injection:** L'aggressore aggiunge esempi errati o fuorvianti al set di training. Questi esempi sono spesso progettati per sembrare normali a un'ispezione superficiale, ma sono stati attentamente elaborati per interrompere il processo di apprendimento.
- **Training:** Il modello ML si allena su questo set di dati manipolato e sviluppa comprensioni distorte dei pattern di dati.
- **Deployment:** Una volta distribuito il modello, l'addestramento corrotto porta a un processo decisionale imperfetto o a vulnerabilità prevedibili che l'aggressore può sfruttare.

L'impatto dell'avvelenamento dei dati si estende oltre gli errori di classificazione o i cali di accuratezza. In applicazioni critiche come l'assistenza sanitaria, tali alterazioni possono portare a significativi problemi di fiducia e sicurezza (Marulli, Marrone, e Verde 2022). Più avanti, discuteremo alcuni casi di studio di questi problemi.

Esistono sei categorie principali di avvelenamento dei dati (Oprea, Singhal, e Vassilev 2022):

- **Attacchi alla Disponibilità:** Questi attacchi mirano a compromettere la funzionalità complessiva di un modello. Fanno sì che classifichi erroneamente la maggior parte dei campioni di test, rendendo il modello inutilizzabile per applicazioni pratiche. Un esempio è il "label flipping", in cui le etichette di una classe specifica e mirata vengono sostituite con etichette di una classe diversa.
- **Attacchi Mirati:** A differenza degli attacchi alla disponibilità, gli attacchi mirati mirano a compromettere un piccolo numero di campioni di test. Quindi, l'effetto è localizzato a un numero limitato di classi, mentre il modello mantiene lo stesso livello originale di accuratezza per la maggior parte delle classi. La natura mirata dell'attacco richiede che l'aggressore conosca le classi del modello, rendendo più difficile il rilevamento di questi attacchi.
- **Attacchi Backdoor:** In questi attacchi, un avversario prende di mira pattern specifici nei dati. L'aggressore introduce una backdoor (un trigger o un pattern nascosto e dannoso) nei dati di training, ad esempio manipolando determinate feature nei dati strutturati o manipolando un pattern di pixel in una posizione fissa. Ciò fa sì che il modello associ il pattern dannoso a etichette specifiche. Di conseguenza, quando il

modello incontra campioni di prova che contengono un pattern dannoso, effettua previsioni false.

- **Attacchi di Sotto-popolazione:** Gli aggressori scelgono selettivamente di compromettere un sottoinsieme dei campioni di test mantenendo l'accuratezza sul resto dei campioni. Questi attacchi si possono pensare come una combinazione di attacchi di disponibilità e mirati: eseguire attacchi di disponibilità (degrado delle prestazioni) nell'ambito di un sottoinsieme mirato. Sebbene gli attacchi di sottopopolazione possano sembrare molto simili agli attacchi mirati, i due presentano chiare differenze:
- **Scope:** Mentre gli attacchi mirati prendono di mira un set selezionato di campioni, gli attacchi di sotto-popolazione prendono di mira una sotto-popolazione generale con rappresentazioni di caratteristiche simili. Ad esempio, in un attacco mirato, un aggressore inserisce immagini manipolate di un cartello di avvertimento di "dosso" (con perturbazioni o pattern accuratamente studiati), che fanno sì che un'auto autonoma non riesca a riconoscere tale cartello e rallenti. D'altro canto, manipolare tutti i campioni di persone con accento britannico in modo che un modello di riconoscimento vocale classifichi erroneamente il discorso di una persona britannica è un esempio di attacco di sotto-popolazione.
- **Conoscenza:** Mentre gli attacchi mirati richiedono un alto grado di familiarità con i dati, gli attacchi alla sotto-popolazione richiedono una conoscenza meno approfondita per essere efficaci.

Le caratteristiche del data poisoning includono:

Manipolazioni sottili e difficili da rilevare dei dati di training: Il data poisoning spesso comporta manipolazioni sottili dei dati di training che sono attentamente studiate per essere difficili da rilevare tramite un'ispezione casuale. Gli aggressori impiegano tecniche sofisticate per garantire che i campioni avvelenati si fondano perfettamente con i dati legittimi, rendendoli più facili da identificare con un'analisi approfondita. Queste manipolazioni possono mirare a caratteristiche o attributi specifici dei dati, come l'alterazione di valori numerici, la modifica di etichette categoriali o l'introduzione di pattern attentamente progettati. L'obiettivo è influenzare il processo di apprendimento del modello eludendo il rilevamento, consentendo ai dati avvelenati di corrompere sottilmente il comportamento del modello.

Può essere eseguito da insider o aggressori esterni: Gli attacchi di data poisoning possono essere eseguiti da vari attori, tra cui insider malintenzionati con accesso ai dati di training e aggressori esterni che trovano modi per influenzare la raccolta dati o la pipeline di pre-elaborazione. Gli insider rappresentano una minaccia significativa perché spesso hanno accesso privilegiato e conoscenza del sistema, il che consente loro di introdurre dati avvelenati senza destare sospetti. D'altro canto, gli aggressori esterni possono sfruttare le vulnerabilità nell'approvvigionamento dei dati, nelle piattaforme di crowdsourcing o nei processi di aggregazione dei dati per iniettare campioni avvelenati nel set di dati di addestramento. Ciò evidenzia l'importanza di implementare controlli di accesso rigorosi, policy di governance dei dati e meccanismi di monitoraggio per mitigare il rischio di minacce interne e attacchi esterni.

Sfrutta le vulnerabilità nella raccolta e pre-elaborazione dei dati: Gli attacchi di avvelenamento dei dati spesso sfruttano le vulnerabilità nelle fasi di raccolta e pre-elaborazione dei dati della pipeline di apprendimento automatico. Gli aggressori progettano attentamente campioni avvelenati per eludere le comuni tecniche di convalida dei dati, assicurandosi che i dati manipolati rientrino comunque in intervalli accettabili, seguano le

distribuzioni previste o mantengano la coerenza con altre funzionalità. Ciò consente ai dati avvelenati di passare attraverso le fasi di pre-elaborazione dei dati senza essere rilevati. Inoltre, gli attacchi di avvelenamento possono sfruttare le debolezze nella preelaborazione dei dati, come una pulizia dei dati inadeguata, un rilevamento insufficiente di valori anomali o la mancanza di controlli di integrità. Gli aggressori possono anche sfruttare la mancanza di solidi meccanismi di tracciamento della provenienza e della discendenza dei dati per introdurre dati avvelenati senza lasciare una traccia. Per affrontare queste vulnerabilità sono necessarie rigorose tecniche di convalida dei dati, rilevamento delle anomalie e tracciamento della provenienza dei dati per garantire l'integrità e l'affidabilità dei dati di training.

Interrompe il processo di apprendimento e distorce il comportamento del modello: Gli attacchi di avvelenamento dei dati sono progettati per interrompere il processo di apprendimento dei modelli di apprendimento automatico e distorcere il loro comportamento verso gli obiettivi dell'aggressore. I dati avvelenati vengono in genere manipolati con obiettivi specifici, come distorcere il comportamento del modello verso determinate classi, introdurre backdoor o degradare le prestazioni complessive. Queste manipolazioni non sono casuali, ma mirate a ottenere i risultati desiderati dall'aggressore. Introducendo incongruenze nelle etichette, in cui i campioni manipolati hanno etichette che non si allineano con la loro vera natura, gli attacchi di avvelenamento possono confondere il modello durante l'addestramento e portare a previsioni distorte o errate. L'interruzione causata dai dati avvelenati può avere conseguenze di vasta portata, poiché il modello compromesso può prendere decisioni imperfette o mostrare un comportamento indesiderato quando viene distribuito in applicazioni del mondo reale.

Influisce sulle prestazioni, l'equità e l'affidabilità del modello: I dati avvelenati nel dataset di addestramento possono avere gravi implicazioni sulle prestazioni, l'equità e l'affidabilità dei modelli di apprendimento automatico. I dati avvelenati possono degradare l'accuratezza e le prestazioni del modello addestrato, portando a un aumento delle classificazioni errate o degli errori nelle previsioni. Ciò può avere conseguenze significative, soprattutto nelle applicazioni critiche in cui gli output del modello influenzano decisioni importanti. Inoltre, gli attacchi di avvelenamento possono introdurre distorsioni e problemi di equità, facendo sì che il modello prenda decisioni discriminatorie o ingiuste per determinati sottogruppi o classi. Ciò mina le responsabilità etiche e sociali dei sistemi di apprendimento automatico e può perpetuare o amplificare i pregiudizi esistenti. Inoltre, i dati avvelenati erodono l'affidabilità e la credibilità dell'intero sistema di apprendimento automatico. Gli output del modello diventano discutibili e potenzialmente dannosi, portando a una perdita di fiducia nell'integrità del sistema. L'impatto dei dati avvelenati può propagarsi nell'intera pipeline ML, influenzando i componenti downstream e le decisioni che si basano sul modello compromesso. Per affrontare queste preoccupazioni è necessaria una solida governance dei dati, un auditing regolare del modello e un monitoraggio continuo per rilevare e mitigare gli effetti degli attacchi di avvelenamento dei dati.

17.4.2.2 Meccanismi di Avvelenamento dei Dati

Gli attacchi di avvelenamento dei dati possono essere eseguiti tramite vari meccanismi, sfruttando diverse vulnerabilità della pipeline ML. Questi meccanismi consentono agli aggressori di manipolare i dati di training e introdurre campioni dannosi che possono compromettere le prestazioni, l'equità o l'integrità del modello. Comprendere questi meccanismi è fondamentale per sviluppare difese efficaci contro l'avvelenamento dei dati e garantire la robustezza dei sistemi ML. I meccanismi di avvelenamento dei dati possono essere ampiamente categorizzati in base all'approccio dell'aggressore e alla fase della

pipeline ML a cui mirano. Alcuni meccanismi comuni includono la modifica delle etichette dei dati di training, l'alterazione dei valori delle feature, l'iniezione di campioni dannosi accuratamente realizzati, lo sfruttamento delle vulnerabilità di raccolta e pre-elaborazione dei dati, la manipolazione dei dati alla fonte, l'avvelenamento dei dati in scenari di apprendimento online e la collaborazione con addetti ai lavori per manipolare i dati. Ognuno di questi meccanismi presenta sfide uniche e richiede diverse strategie di mitigazione. Ad esempio, rilevare la manipolazione delle etichette può comportare l'analisi della distribuzione delle etichette e l'identificazione delle anomalie (P. Zhou et al. 2018), mentre prevenire la manipolazione delle feature può richiedere tecniche di pre-elaborazione dei dati e rilevamento delle anomalie sicure (Carta et al. 2020). La difesa dalle minacce interne può comportare rigide policy di controllo degli accessi e il monitoraggio dei pattern di accesso ai dati. Inoltre, l'efficacia degli attacchi di avvelenamento dei dati spesso dipende dalla conoscenza del sistema ML da parte dell'attaccante, tra cui l'architettura del modello, gli algoritmi di training e la distribuzione dei dati. Gli aggressori possono utilizzare tecniche di apprendimento automatico avversario o di sintesi dei dati per creare campioni che hanno maggiori probabilità di aggirare il rilevamento e raggiungere i loro obiettivi malevoli.

Modifica delle etichette dei dati di training: Uno dei meccanismi più semplici di avvelenamento dei dati è la modifica delle etichette dei dati di training. In questo approccio, l'aggressore modifica selettivamente le etichette di un sottoinsieme dei campioni di training per fuorviare il processo di apprendimento del modello, come mostrato in Figura 17.23. Ad esempio, in un'attività di classificazione binaria, l'aggressore potrebbe capovolgere le etichette di alcuni campioni positivi in negativi o viceversa. Introducendo tale rumore di etichetta, l'aggressore degrada le prestazioni del modello o fa sì che faccia previsioni errate per istanze target specifiche.

Figura 17.23: Garbage In – Garbage Out. Fonte: Information Matters

Alterazione dei valori delle feature nei dati di training: Un altro meccanismo di avvelenamento dei dati consiste nell'alterare i valori delle caratteristiche dei campioni di training senza modificare le etichette. L'aggressore elabora attentamente i valori delle feature per introdurre specifici pregiudizi o vulnerabilità nel modello. Ad esempio, in

un'attività di classificazione delle immagini, l'aggressore potrebbe aggiungere perturbazioni impercettibili a un sottoinsieme di immagini, facendo sì che il modello apprenda un particolare pattern o associazione. Questo tipo di avvelenamento può creare backdoor o trojan nel modello addestrato, che possono essere attivati da specifici pattern di input.

Iniezione di campioni dannosi accuratamente realizzati: In questo meccanismo, l'aggressore crea campioni dannosi progettati per avvelenare il modello. Questi campioni sono realizzati per avere un impatto specifico sul comportamento del modello, mentre si fondono con i dati di addestramento legittimi. L'aggressore potrebbe utilizzare tecniche come perturbazioni avversarie o sintesi dei dati per generare campioni avvelenati difficili da rilevare. L'aggressore manipola i limiti decisionali del modello iniettando questi campioni dannosi nei dati di addestramento o introducendo classificazioni errate mirate.

Sfruttamento delle vulnerabilità di raccolta e preelaborazione dei dati: Gli attacchi di avvelenamento dei dati possono anche sfruttare le vulnerabilità della pipeline di raccolta e preelaborazione dei dati. Se il processo di raccolta dati non è sicuro o ci sono debolezze nelle fasi di pre-elaborazione dei dati, un aggressore può manipolare i dati prima che raggiungano la fase di addestramento. Ad esempio, se i dati vengono raccolti da fonti non attendibili o ci sono problemi nella pulizia o nell'aggregazione dei dati, un aggressore può introdurre campioni avvelenati o manipolare i dati a proprio vantaggio.

Manipolazione dei dati alla fonte (ad esempio, dati dei sensori): In alcuni casi, gli aggressori possono manipolare i dati alla fonte, come dati dei sensori o dispositivi di input. Manomettendo i sensori o manipolando l'ambiente in cui vengono raccolti i dati, gli aggressori possono introdurre campioni avvelenati o alterare la distribuzione dei dati. Ad esempio, in uno scenario di auto a guida autonoma, un aggressore potrebbe manipolare i sensori o l'ambiente per immettere informazioni fuorvianti nei dati di addestramento, compromettendo la capacità del modello di prendere decisioni sicure e affidabili.

Avvelenamento dei dati in scenari di apprendimento online: Gli attacchi di avvelenamento dei dati possono anche colpire sistemi ML che impiegano l'apprendimento online, in cui il modello viene costantemente aggiornato con nuovi dati in tempo reale. In tali scenari, un aggressore può gradualmente iniettare campioni avvelenati nel tempo, manipolando lentamente il comportamento del modello. I sistemi di apprendimento online sono particolarmente vulnerabili all'avvelenamento dei dati perché si adattano ai nuovi dati senza una convalida estesa, rendendo più facile per gli aggressori introdurre campioni dannosi, come mostrato in Figura 17.24.

Figura 17.24: Attacco di Avvelenamento dei Dati. Fonte: Sikandar

Collaborazione con addetti ai lavori per manipolare i dati: A volte, gli attacchi di avvelenamento dei dati possono comportare la collaborazione con addetti ai lavori con accesso ai dati di training. Gli addetti ai lavori malintenzionati, come dipendenti o provider di dati, possono manipolare i dati prima che vengano utilizzati per addestrare il modello. Le minacce interne sono particolarmente difficili da rilevare e prevenire, poiché gli aggressori hanno un accesso legittimo ai dati e possono elaborare attentamente la strategia di avvelenamento per eludere il rilevamento.
Questi sono i meccanismi chiave dell'avvelenamento dei dati nei sistemi ML. Gli aggressori spesso impiegano questi meccanismi per rendere i loro attacchi più efficaci e difficili da rilevare. Il rischio di attacchi di avvelenamento dei dati aumenta man mano che i sistemi ML diventano sempre più complessi e si basano su set di dati più grandi provenienti da fonti diverse. La difesa dall'avvelenamento dei dati richiede un approccio poliedrico. I professionisti ML e i progettisti di sistemi devono essere consapevoli dei vari meccanismi di avvelenamento dei dati e adottare un approccio completo alla sicurezza dei dati e alla resilienza del modello. Ciò include la raccolta dati sicura, la convalida dati robusta e il monitoraggio continuo delle prestazioni del modello. L'implementazione di pratiche di raccolta dati e pre-elaborazione sicure è fondamentale per prevenire l'avvelenamento dei dati alla fonte. Le tecniche di convalida dati e rilevamento anomalie possono anche aiutare a identificare e mitigare potenziali tentativi di avvelenamento. Il monitoraggio delle prestazioni del modello per segnali di avvelenamento dei dati è inoltre essenziale per rilevare e rispondere prontamente agli attacchi.

17.4.2.3 Impatto sui Sistemi ML

Gli attacchi di avvelenamento dei dati possono avere gravi ripercussioni sui sistemi ML, compromettendone le prestazioni, l'affidabilità e la credibilità. L'impatto dell'avvelenamento dei dati può manifestarsi in vari modi, a seconda degli obiettivi dell'aggressore e del meccanismo specifico utilizzato. Analizziamo in dettaglio ciascuno dei potenziali impatti.

Degrado delle prestazioni del modello: Uno degli impatti principali dell'avvelenamento dei dati è il degrado delle prestazioni complessive del modello. Manipolando i dati di training, gli aggressori possono introdurre rumore, distorsioni o incongruenze che ostacolano la capacità del modello di apprendere pattern accurati e fare previsioni affidabili. Ciò può ridurre accuratezza, precisione, richiamo o altre metriche delle prestazioni. Il degrado delle prestazioni del modello può avere conseguenze significative, soprattutto in applicazioni critiche come sanità, finanza o sicurezza, dove l'affidabilità delle previsioni è fondamentale.

Errore di classificazione di target specifici: Gli attacchi di avvelenamento dei dati possono anche essere progettati per far sì che il modello classifichi in modo errato istanze target specifiche. Gli aggressori possono introdurre campioni avvelenati realizzati con cura simili alle istanze target, portando il modello ad apprendere associazioni errate. Ciò può comportare che il modello classifichi in modo errato le istanze target in modo coerente, anche se funziona bene su altri input. Tale errata classificazione mirata può avere gravi conseguenze, come far sì che un sistema di rilevamento malware trascuri file dannosi specifici o portare a una diagnosi errata in un'applicazione di imaging medico.

Backdoor e trojan nei modelli addestrati: L'avvelenamento dei dati può introdurre backdoor o trojan nel modello addestrato. Le backdoor sono funzionalità nascoste che consentono agli aggressori di innescare comportamenti specifici o bypassare i normali meccanismi di autenticazione. D'altro canto, i trojan sono componenti dannosi insinuati nel modello che possono attivare specifici pattern di input. Avvelenando i dati di training, gli aggressori possono creare modelli che sembrano funzionare normalmente ma contengono vulnerabilità nascoste che possono essere sfruttate in seguito. Backdoor e trojan possono

compromettere l'integrità e la sicurezza del sistema ML, consentendo agli aggressori di ottenere accesso non autorizzato, manipolare previsioni o esfiltrare informazioni sensibili.

Risultati del modello distorti o ingiusti: Gli attacchi di avvelenamento dei dati possono introdurre distorsioni o ingiustizie nelle previsioni del modello. Manipolando la distribuzione dei dati di training o iniettando campioni con distorsioni specifiche, gli aggressori possono far sì che il modello apprenda e perpetui pattern discriminatori. Ciò può portare a un trattamento ingiusto di determinati gruppi o individui in base ad attributi sensibili come razza, genere o età. I modelli distorti possono avere gravi implicazioni sociali, rafforzando le disuguaglianze e le pratiche discriminatorie esistenti. Garantire l'equità e mitigare i pregiudizi è fondamentale per creare sistemi ML affidabili ed etici.

Aumento di falsi positivi o falsi negativi: L'avvelenamento dei dati può anche influire sulla capacità del modello di identificare correttamente istanze positive o negative, portando a un aumento di falsi positivi o falsi negativi. I falsi positivi si verificano quando il modello identifica erroneamente un'istanza negativa come positiva, mentre i falsi negativi si verificano quando un'istanza positiva viene classificata erroneamente come negativa. Le conseguenze dell'aumento di falsi positivi o falsi negativi possono essere significative a seconda dell'applicazione. Ad esempio, in un sistema di rilevamento delle frodi, un elevato numero di falsi positivi può portare a indagini non necessarie e frustrazione dei clienti, mentre un elevato numero di falsi negativi può consentire che le attività fraudolente passino inosservate.

Affidabilità e fiducia del sistema compromesse: Gli attacchi di avvelenamento dei dati possono minare l'affidabilità e la fiducia complessiva dei sistemi ML. Quando i modelli vengono addestrati su dati contaminati, le loro previsioni diventano inaffidabili e inaffidabili. Ciò può erodere la fiducia dell'utente nel sistema e portare a una perdita di fiducia nelle decisioni prese dal modello. Nelle applicazioni critiche in cui si fa affidamento sui sistemi ML per il processo decisionale, come veicoli autonomi o diagnosi mediche, l'affidabilità compromessa può avere gravi conseguenze, mettendo a rischio vite e proprietà.
Per affrontare l'impatto dell'avvelenamento dei dati è necessario un approccio proattivo alla sicurezza dei dati, ai test dei modelli e al monitoraggio. Le organizzazioni devono implementare misure robuste per garantire l'integrità e la qualità dei dati di training, impiegare tecniche per rilevare e mitigare i tentativi di avvelenamento e monitorare costantemente le prestazioni e il comportamento dei modelli distribuiti. La collaborazione tra professionisti ML, esperti di sicurezza e specialisti di dominio è essenziale per sviluppare strategie complete per prevenire e rispondere agli attacchi di avvelenamento dei dati.

17.4.2.3.1 Caso di Studio

È interessante notare che gli attacchi di "data poisoning" non sono sempre dannosi (Shan et al. 2023). Nightshade, uno strumento sviluppato da un team guidato dal professor Ben Zhao presso l'Università di Chicago, utilizza l'avvelenamento dei dati per aiutare gli artisti a proteggere la propria arte da scraping e violazioni del copyright da parte di modelli di IA generativa. Gli artisti possono utilizzare lo strumento per apportare modifiche impercettibili alle proprie immagini prima di caricarle online, come mostrato in Figura 17.25.

Figura 17.25: Campioni di dati avvelenati con etichette sbagliateriguardanti coppie testo/immagine non corrispondenti. Fonte: Shan

Sebbene queste modifiche siano impercettibili all'occhio umano, possono compromettere significativamente le prestazioni dei modelli di intelligenza artificiale generativa quando vengono incorporate nei dati di addestramento. I modelli generativi possono essere manipolati per generare allucinazioni e immagini strane. Ad esempio, con solo 300 immagini avvelenate, i ricercatori dell'Università di Chicago potrebbero ingannare l'ultimo modello Stable Diffusion per generare immagini di cani che sembrano gatti o immagini di mucche quando vengono richieste le auto.

Man mano che aumenta il numero di immagini avvelenate su Internet, le prestazioni dei modelli che utilizzano dati acquisiti peggioreranno in modo esponenziale. In primo luogo, i dati avvelenati sono difficili da rilevare e richiedono l'eliminazione manuale. In secondo luogo, il "veleno" si diffonde rapidamente ad altre etichette perché i modelli generativi si basano su connessioni tra parole e concetti mentre generano immagini. Quindi un'immagine avvelenata di una "macchina" potrebbe diffondersi in immagini generate associate a parole come "camion", "treno", "autobus", ecc.

D'altra parte, questo strumento può essere utilizzato in modo dannoso e può influenzare le applicazioni legittime dei modelli generativi. Ciò dimostra la natura molto impegnativa e innovativa degli attacchi di apprendimento automatico.

Figura 17.26 mostra gli effetti di diversi livelli di avvelenamento dei dati (50 campioni, 100 campioni e 300 campioni di immagini avvelenate) sulla generazione di immagini in diverse categorie. Notare come le immagini iniziano a deformarsi e deviare dalla categoria desiderata. Ad esempio, dopo 300 campioni di avvelenamento, una richiesta di un'auto genera una mucca.

Figura 17.26: Avvelenamento dei Dati. Fonte: Shan et al. (2023))

Esercizio 17.3: Attacchi Avvelenati

Preparatevi a esplorare il lato oscuro della sicurezza dell'IA! In questo Colab, impareremo cos'è l'avvelenamento dei dati, ovvero come dati errati possono ingannare i modelli di IA e fargli prendere decisioni sbagliate. Ci concentreremo su un attacco reale contro una Support Vector Machine (SVM), osservando come cambia il comportamento dell'IA sotto attacco. Questo esercizio pratico metterà in evidenza perché proteggere i sistemi di IA è fondamentale, soprattutto man mano che diventano più integrati nelle nostre vite. Pensare come un hacker, comprendere la vulnerabilità e fare brainstorming su come difendere i sistemi di IA!

Open in Colab

17.4.3 Distribution Shift

17.4.3.1 Definizione e Caratteristiche

La "distribution shift" [slittamento della distribuzione] si riferisce al fenomeno in cui la distribuzione dei dati incontrata da un modello ML durante la distribuzione (inferenza) differisce dalla distribuzione su cui è stato addestrato, come mostrato in Figura 17.27. Questo non è tanto un attacco quanto il fatto che la robustezza del modello varierà nel tempo. In altre parole, le proprietà statistiche, i pattern o le ipotesi sottostanti dei dati possono cambiare tra le fasi di addestramento e di test.

Figura 17.27: Le parentesi graffe racchiudono la "distribution shift" tra gli ambienti. Qui, z sta per la caratteristica spuria e y sta per la classe dell'etichetta. Fonte: Xin

Le caratteristiche principali della "distribution shift" includono:

Discordanza di dominio: I dati di input durante l'inferenza provengono da un dominio o una distribuzione diversi rispetto ai dati di addestramento. Quando i dati di input durante l'inferenza provengono da un dominio o una distribuzione diversi dai dati di training, possono influenzare significativamente le prestazioni del modello. Questo perché il modello ha imparato pattern e relazioni specifici del dominio di training e, se applicati a un dominio diverso, tali pattern appresi potrebbero non essere validi. Ad esempio, si consideri un modello di analisi del sentiment addestrato sulle recensioni di film. Supponiamo che questo modello venga applicato per analizzare il sentiment nei tweet. In tal caso, potrebbe aver bisogno di aiuto per classificare accuratamente il sentiment perché la lingua, la grammatica e il contesto dei tweet possono differire dalle recensioni dei film. Questa discrepanza di

dominio può causare scarse prestazioni e previsioni inaffidabili, limitando l'utilità pratica del modello.

Deriva temporale: La distribuzione dei dati si evolve, portando a uno spostamento graduale o improvviso nelle caratteristiche di input. La deriva temporale è importante perché i modelli ML vengono spesso distribuiti in ambienti dinamici in cui la distribuzione dei dati può cambiare nel tempo. Se il modello non viene aggiornato o adattato a questi cambiamenti, le sue prestazioni possono gradualmente peggiorare. Ad esempio, i pattern e i comportamenti associati alle attività fraudolente possono evolversi in un sistema di rilevamento delle frodi man mano che i truffatori adattano le loro tecniche. Se il modello non viene riqualificato o aggiornato per catturare questi nuovi pattern, potrebbe non riuscire a rilevare efficacemente nuovi tipi di frode. La deriva temporale può portare a un calo dell'accuratezza e dell'affidabilità del modello nel tempo, rendendo cruciale il monitoraggio e l'affronto di questo tipo di spostamento della distribuzione.

Cambiamenti contestuali: Il contesto del modello ML può variare, determinando diverse distribuzioni di dati in base a fattori quali posizione, comportamento dell'utente o condizioni ambientali. I cambiamenti contestuali sono importanti perché i modelli ML vengono spesso distribuiti in vari contesti o ambienti che possono avere diverse distribuzioni di dati. Se il modello non riesce a generalizzarsi bene a questi diversi contesti, le sue prestazioni potrebbero deteriorarsi. Ad esempio, si consideri un modello di visione artificiale addestrato per riconoscere oggetti in un ambiente di laboratorio controllato. Quando distribuito in un contesto reale, fattori quali condizioni di illuminazione, angoli della telecamera o confusione sullo sfondo possono variare in modo significativo, determinando una "distribution shift". Se il modello è robusto a questi cambiamenti contestuali, potrebbe essere in grado di riconoscere accuratamente gli oggetti nel nuovo ambiente, limitandone l'utilità pratica.

Dati di addestramento non rappresentativi: I dati di addestramento potrebbero catturare solo parzialmente la variabilità e la diversità dei dati del mondo reale riscontrati durante la distribuzione. I dati di training non rappresentativi possono portare a modelli parziali o distorti che funzionano male sui dati del mondo reale. Supponiamo che i dati di training debbano catturare adeguatamente la variabilità e la diversità dei dati del mondo reale. In tal caso, il modello potrebbe apprendere pattern specifici del set di training, ma deve essere meglio generalizzato a dati nuovi e mai visti. Ciò può comportare scarse prestazioni, previsioni parziali e limitata applicabilità del modello. Ad esempio, se un modello di riconoscimento facciale viene addestrato principalmente su immagini di individui di uno specifico gruppo demografico, potrebbe avere difficoltà a riconoscere accuratamente i volti di altri gruppi demografici quando viene distribuito in un contesto reale. Garantire che i dati di training siano rappresentativi e diversificati è fondamentale per creare modelli che possano essere generalizzati bene a scenari del mondo reale.
La "distribution shift" può manifestarsi in varie forme, come:

Covariate shift: La distribuzione delle feature di input (covariate) cambia mentre la distribuzione condizionale della variabile target dato l'input rimane la stessa. La "covariate shift" è importante perché può influire sulla capacità del modello di fare previsioni accurate quando le feature di input (covariate) differiscono tra i dati di training e quelli di test. Anche se la relazione tra le feature di input e la variabile target rimane la stessa, un cambiamento nella distribuzione delle feature di input può influire sulle prestazioni del modello. Ad esempio, si consideri un modello addestrato per prevedere i prezzi delle case in base a caratteristiche come la metratura, il numero di camere da letto e la posizione. Supponiamo che la distribuzione di queste caratteristiche nei dati di test differisca significativamente dai dati di training (ad esempio, i dati di test contengono case con una metratura molto più ampia). In

tal caso, le previsioni del modello potrebbero diventare meno accurate. È importante tenere conto dei "covariate shift" per garantire la robustezza e l'affidabilità del modello quando viene applicato a nuovi dati.

Concept drift: La relazione tra le feature di input e la variabile target cambia nel tempo, alterando il concetto sottostante che il modello sta cercando di apprendere, come mostrato in Figura 17.28. Il "concept drift" è importante perché indica cambiamenti nella relazione fondamentale tra le feature di input e la variabile target nel tempo. Quando il concetto sottostante che il modello sta cercando di apprendere cambia, le sue prestazioni possono deteriorarsi se non vengono adattate al nuovo concetto. Ad esempio, in un modello di previsione dell'abbandono dei clienti, i fattori che influenzano l'abbandono dei clienti possono evolversi a causa delle condizioni di mercato, delle offerte della concorrenza o delle preferenze dei clienti. Se il modello non viene aggiornato per catturare questi cambiamenti, le sue previsioni potrebbero diventare meno accurate e irrilevanti. Rilevare e adattarsi al "concept drift" è fondamentale per mantenere l'efficacia del modello e l'allineamento con i concetti del mondo reale in evoluzione.

Figura 17.28: La deriva concettuale si riferisce a un cambiamento nei pattern e nelle relazioni dei dati nel tempo. Fonte: Evidently AI

Generalizzazione del dominio: Il modello deve generalizzare a domini o distribuzioni invisibili non presenti durante l'addestramento. La generalizzazione di dominio è importante perché consente di applicare i modelli ML a nuovi domini mai visti senza richiedere un'ampia riqualificazione o adattamento. Negli scenari del mondo reale, i dati di addestramento che coprono tutti i possibili domini o distribuzioni che il modello può incontrare sono spesso irrealizzabili. Le tecniche di generalizzazione di dominio mirano ad apprendere caratteristiche o modelli invarianti al dominio che possono essere generalizzati bene a nuovi domini. Ad esempio, si consideri un modello addestrato per classificare immagini di animali. Se il modello può apprendere caratteristiche invarianti a diversi sfondi, condizioni di illuminazione o pose, può essere generalizzato bene per classificare animali in nuovi ambienti mai visti. La generalizzazione del dominio è fondamentale per creare modelli che possono essere distribuiti in contesti reali diversi e in continua evoluzione.
La presenza di un "distribution shift" può avere un impatto significativo sulle prestazioni e l'affidabilità dei modelli ML, poiché i modelli potrebbero aver bisogno di aiuto per

generalizzare bene alla nuova distribuzione dei dati. Rilevare e adattarsi ai “distribution shift” è fondamentale per garantire la robustezza e l’utilità pratica dei sistemi ML negli scenari del mondo reale.

17.4.3.2 Meccanismi delle Distribution Shift

I meccanismi della “distribution shift”, come cambiamenti nelle fonti dei dati, evoluzione temporale, variazioni specifiche del dominio, bias di selezione, cicli di feedback e manipolazioni avversarie, sono importanti da comprendere perché aiutano a identificarne le cause. Comprendendo questi meccanismi, i professionisti possono sviluppare strategie mirate per mitigarne l’impatto e migliorare la robustezza del modello. Ecco alcuni meccanismi comuni:

Figura 17.29: Evoluzione temporale. Fonte: Białek

Cambiamenti nelle fonti di dati: Possono verificarsi cambiamenti di distribuzione quando le fonti di dati utilizzate per l’addestramento e l’inferenza sono diverse. Ad esempio, se un modello viene addestrato sui dati di un sensore ma distribuito sui dati di un altro sensore con caratteristiche diverse, può portare a un “distribution shift”.

Evoluzione temporale: Nel tempo, la distribuzione dei dati sottostante può evolversi a causa di cambiamenti nel comportamento dell’utente, dinamiche di mercato o altri fattori temporali. Ad esempio, in un sistema di raccomandazione, le preferenze dell’utente possono cambiare nel tempo, portando a un “distribution shift” nei dati di input, come mostrato in Figura 17.29.

Variazioni specifiche del dominio: Domini o contesti diversi possono avere distribuzioni di dati distinte. Un modello addestrato sui dati di un dominio può generalizzare bene con un altro dominio solo con tecniche di adattamento appropriate. Ad esempio, un modello di classificazione delle immagini addestrato su scene di interni potrebbe avere difficoltà se applicato a scene in esterno.

Bias di selezione: Un “Distribution shift” può derivare da un bias di selezione durante la raccolta o il campionamento dei dati. Se i dati di training non rappresentano la popolazione reale o determinati sottogruppi sono sovrarappresentati o sottorappresentati, si può arrivare a una mancata corrispondenza tra le distribuzioni di training e di test.

Cicli di feedback: In alcuni casi, le previsioni o le azioni intraprese da un modello ML possono influenzare la futura distribuzione dei dati. Ad esempio, in un sistema di prezzi

dinamici, i prezzi stabiliti dal modello possono influire sul comportamento dei clienti, determinando uno spostamento nella distribuzione dei dati nel tempo.

Manipolazioni avversarie: Gli avversari possono manipolare intenzionalmente i dati di input per creare uno spostamento della distribuzione e ingannare il modello ML. Introducendo perturbazioni attentamente studiate o generando campioni fuori distribuzione, gli aggressori possono sfruttare le vulnerabilità del modello e fargli fare previsioni errate.
Comprendere i meccanismi del "distribution shift" è importante per sviluppare strategie efficaci per rilevare e mitigare il suo impatto sui sistemi ML. Identificando le fonti e le caratteristiche dello spostamento, i professionisti possono progettare tecniche appropriate, come l'adattamento del dominio, l'apprendimento tramite trasferimento o l'apprendimento continuo, per migliorare la robustezza e le prestazioni del modello in caso di cambiamenti distributivi.

17.4.3.3 Impatto sui Sistemi ML

I "distribution shift" possono avere un impatto negativo significativo sulle prestazioni e l'affidabilità dei sistemi ML. Ecco alcuni modi chiave in cui il "distribution shift" può influenzare i modelli ML:

Prestazioni predittive degradate: Quando la distribuzione dei dati riscontrata durante l'inferenza differisce dalla distribuzione di training, l'accuratezza predittiva del modello può deteriorarsi. Il modello potrebbe aver bisogno di aiuto per generalizzare bene i nuovi dati, il che porta a un aumento degli errori e a prestazioni non ottimali.

Affidabilità e attendibilità ridotte: Il "distribution shift" può compromettere l'affidabilità e l'attendibilità dei modelli ML. Se le previsioni del modello diventano inaffidabili o incoerenti a causa dello spostamento, gli utenti potrebbero perdere fiducia negli output del sistema, il che porta a un potenziale uso improprio o non uso del modello.

Predizioni distorte: Lo spostamento di distribuzione può introdurre "bias" [distorsioni] nelle previsioni del modello. Se i dati di training non rappresentano la distribuzione nel mondo reale o alcuni sottogruppi sono sottorappresentati, il modello potrebbe fare previsioni distorte che discriminano determinati gruppi o perpetuano pregiudizi sociali.

Maggiore incertezza e rischio: Lo spostamento della distribuzione introduce ulteriore incertezza e rischio nel sistema ML. Il comportamento e le prestazioni del modello potrebbero diventare meno prevedibili, rendendo difficile valutarne l'affidabilità e l'idoneità per applicazioni critiche. Questa incertezza può portare a maggiori rischi operativi e potenziali guasti.

Sfide di adattabilità: I modelli ML addestrati su una distribuzione dati specifica potrebbero aver bisogno di aiuto per adattarsi ad ambienti mutevoli o nuovi domini. La mancanza di adattabilità può limitare l'utilità e l'applicabilità del modello in scenari reali dinamici in cui la distribuzione dei dati si evolve.

Difficoltà di manutenzione e aggiornamento: Il "distribution shift" può complicare la manutenzione e l'aggiornamento dei modelli ML. Man mano che la distribuzione dei dati cambia, il modello potrebbe richiedere frequenti riqualificazioni o ottimizzazioni per mantenere le sue prestazioni. Ciò può richiedere molto tempo e risorse, soprattutto se il cambiamento avviene rapidamente o continuamente.

Vulnerabilità agli attacchi avversari: Il "distribution shift" può rendere i modelli ML più vulnerabili agli attacchi avversari. Gli avversari possono sfruttare la sensibilità del modello ai cambiamenti distributivi creando esempi avversari al di fuori della distribuzione di

addestramento, facendo sì che il modello faccia previsioni errate o si comporti in modo inaspettato.
Per mitigare l'impatto dei "distribution shift", è fondamentale sviluppare sistemi ML robusti che rilevino e si adattino ai cambiamenti delle distribuzioni. Tecniche come l'adattamento del dominio, l'apprendimento tramite trasferimento e l'apprendimento continuo possono aiutare a migliorare la capacità di generalizzazione del modello su diverse distribuzioni. Il monitoraggio, il test e l'aggiornamento del modello ML sono inoltre necessari per garantirne le prestazioni e l'affidabilità durante i "distribution shift".

17.4.4 Rilevamento e Mitigazione

17.4.4.1 Attacchi Avversari

Come si ricorderà da quanto sopra, gli attacchi avversari rappresentano una minaccia significativa per la robustezza e l'affidabilità dei sistemi ML. Questi attacchi comportano la creazione di input attentamente progettati, noti come "esempi avversari", per ingannare i modelli ML e fargli fare previsioni errate. Per proteggere i sistemi ML dagli attacchi avversari, è fondamentale sviluppare tecniche efficaci per rilevare e mitigare queste minacce.

17.4.4.1.1 Tecniche di Rilevamento degli Esempi Avversari

Il rilevamento degli esempi avversari è la prima linea di difesa contro gli attacchi avversari. Sono state proposte diverse tecniche per identificare e segnalare input sospetti che potrebbero essere avversari.
I metodi statistici mirano a rilevare gli esempi avversari analizzando le proprietà statistiche dei dati di input. Questi metodi spesso confrontano la distribuzione dei dati di input con una di riferimento, come quella dei dati di training o una nota distribuzione benigna. Tecniche come il test Kolmogorov-Smirnov (Berger e Zhou 2014) o il test Anderson-Darling possono essere utilizzate per misurare la discrepanza tra le distribuzioni e segnalare gli input che si discostano in modo significativo dalla distribuzione prevista.
Kernel density estimation (KDE) è una tecnica non parametrica utilizzata per stimare la funzione di densità di probabilità di un set di dati. Nel contesto del rilevamento di esempi avversari, KDE può essere utilizzato per stimare la densità di esempi benigni nello spazio di input. Gli esempi avversari spesso si trovano in regioni a bassa densità e possono essere rilevati confrontando la loro densità stimata con una soglia. Gli input con una densità stimata al di sotto della soglia vengono segnalati come potenziali esempi avversari.
Un'altra tecnica è la compressione delle feature (Panda, Chakraborty, e Roy 2019), che riduce la complessità dello spazio di input applicando la riduzione della dimensionalità o la discretizzazione. L'idea alla base della compressione delle feature è che gli esempi avversari spesso si basano su piccole perturbazioni impercettibili che possono essere eliminate o ridotte tramite queste trasformazioni. Le incongruenze possono essere rilevate confrontando le previsioni del modello sull'input originale e sull'input compresso, indicando la presenza di esempi avversari.
Le tecniche di stima dell'incertezza del modello mirano a quantificare la fiducia o l'incertezza associata alle previsioni di un modello. Gli esempi avversari spesso sfruttano regioni di elevata incertezza nel confine di decisione del modello. Stimando l'incertezza utilizzando tecniche come reti neurali bayesiane, stima dell'incertezza basata su dropout o metodi di ensemble, gli input con elevata incertezza possono essere contrassegnati come potenziali esempi avversari.

17.4.4.1.2 Strategie di Difesa Avversarie

Una volta rilevati gli esempi avversari, possono essere impiegate varie strategie di difesa per mitigarne l'impatto e migliorare la robustezza dei modelli ML.

L'addestramento avversario è una tecnica che prevede l'aumento dei dati di addestramento con esempi avversari e il riaddestramento del modello su questo set di dati aumentato. Esporre il modello a esempi avversari durante l'addestramento gli insegna a classificarli correttamente e diventa più robusto agli attacchi avversari. L'addestramento avversario può essere eseguito utilizzando vari metodi di attacco, come il Fast Gradient Sign Method (FGSM) o il Projected Gradient Descent (PGD) (Madry et al. 2017).
La distillazione difensiva (Papernot et al. 2016) è una tecnica che addestra un secondo modello (il modello studente) per imitare il comportamento di quello originale (il modello insegnante). Il modello studente viene addestrato sulle etichette soft prodotte dal modello insegnante, che sono meno sensibili alle piccole perturbazioni. L'utilizzo del modello studente per l'inferenza può ridurre l'impatto delle perturbazioni avversarie, poiché il modello studente impara a generalizzare meglio ed è meno sensibile al rumore avversario.
Le tecniche di pre-elaborazione e trasformazione dell'input mirano a rimuovere o mitigare l'effetto delle perturbazioni avversarie prima di alimentare l'input nel modello ML. Queste tecniche includono la rimozione del rumore dalle immagini, la compressione JPEG, il ridimensionamento casuale, il padding o l'applicazione di trasformazioni casuali ai dati di input. Riducendo l'impatto delle perturbazioni avversarie, questi passaggi di pre-elaborazione possono aiutare a migliorare la robustezza del modello agli attacchi avversari.
I metodi ensemble combinano più modelli per fare previsioni più robuste. L'ensemble può ridurre l'impatto degli attacchi avversari utilizzando un set diversificato di modelli con diverse architetture, dati di training o iperparametri. Esempi avversari che ingannano un modello potrebbero non ingannare gli altri nell'insieme, portando a previsioni più affidabili e robuste.
Le tecniche di diversificazione del modello, come l'utilizzo di diverse tecniche di pre-elaborazione o rappresentazioni delle caratteristiche per ogni modello nell'insieme, possono migliorare ulteriormente la robustezza.

17.4.4.1.3 Valutazione e Test della Robustezza

Condurre valutazioni e test approfonditi per valutare l'efficacia delle tecniche di difesa avversarie e misurare la robustezza dei modelli ML.
Le metriche di robustezza avversaria quantificano la resilienza del modello agli attacchi avversari. Queste metriche possono includere l'accuratezza del modello sugli esempi avversari, la distorsione media richiesta per ingannare il modello o le prestazioni del modello in base a diversi livelli di attacco. Confrontando queste metriche tra diversi modelli o tecniche di difesa, i professionisti possono valutare e confrontare i loro livelli di robustezza.
I benchmark e i set di dati standardizzati per gli attacchi avversari forniscono una base comune per valutare e confrontare la robustezza dei modelli ML. Questi benchmark includono set di dati con esempi avversari pre-generati e strumenti e framework per generare attacchi avversari. Esempi di benchmark di attacchi avversari popolari includono i set di dati MNIST-C, CIFAR-10-C e ImageNet-C (Hendrycks e Dietterich 2019), che contengono versioni corrotte o perturbate dei set di dati originali.
I professionisti possono sviluppare sistemi ML più robusti e resilienti sfruttando queste tecniche di rilevamento di esempi avversari, strategie di difesa e metodi di valutazione della robustezza. Tuttavia, è importante notare che la robustezza avversaria è un'area di ricerca in corso e nessuna tecnica singola fornisce una protezione completa contro tutti i tipi di attacchi avversari. Un approccio completo che combina più meccanismi di difesa e test regolari è essenziale per mantenere la sicurezza e l'affidabilità dei sistemi ML di fronte alle minacce avversarie in evoluzione.

17.4.4.2 Avvelenamento dei Dati

Si ricorda che il data poisoning è un attacco che prende di mira l'integrità dei dati di training utilizzati per creare modelli ML. Manipolando o corrompendo i dati di training, gli aggressori

possono influenzare il comportamento del modello e fargli fare previsioni errate o eseguire azioni indesiderate. Rilevare e mitigare gli attacchi di data poisoning è fondamentale per garantire l'affidabilità e la sicurezza dei sistemi ML, come mostrato in Figura 17.30.

Figura 17.30: Iniezione di dati dannosi. Fonte: Li

17.4.4.2.1 Tecniche di rilevamento delle anomalie per identificare i Dati Avvelenati

I metodi di rilevamento statistico degli outlier identificano i dati che si discostano in modo significativo dalla maggior parte. Questi metodi presuppongono che le istanze di dati avvelenati siano probabilmente "outlier" statistici". Tecniche come il Metodo Z-score, il Metodo di Tukey o la Distanza di Mahalanobis possono essere utilizzate per misurare la deviazione di ciascun punto dati dalla tendenza centrale del set di dati. I dati che superano una soglia predefinita vengono contrassegnati come potenziali valori anomali e considerati sospetti di avvelenamento dei dati.

I metodi basati sul clustering raggruppano dati simili in base alle loro caratteristiche o attributi. Il presupposto è che le istanze di dati avvelenate possano formare cluster distinti o trovarsi lontano dai normali cluster di dati. Applicando algoritmi di clustering come K-means, DBSCAN o clustering gerarchico, è possibile identificare cluster anomali o dati che non appartengono a nessun cluster. Queste istanze anomale vengono poi trattate come dati potenzialmente avvelenati.

Gli autoencoder sono reti neurali addestrate per ricostruire i dati di input da una rappresentazione compressa, come mostrato in Figura 17.31. Possono essere utilizzati per il rilevamento di anomalie apprendendo i pattern normali nei dati e identificando le istanze che si discostano da essi. Durante l'addestramento, l'autoencoder viene addestrato su dati puliti e non avvelenati. Al momento dell'inferenza, viene calcolato l'errore di ricostruzione per ogni dato. I dati con errori di ricostruzione elevati sono considerati anomali e potenzialmente avvelenati, poiché non sono conformi ai pattern normali appresi.

Figura 17.31: Autoencoder. Fonte: Dertat

17.4.4.2.2 Tecniche di Sanificazione e Preelaborazione dei Dati

L'avvelenamento dei dati può essere evitato pulendo i dati, il che implica l'identificazione e la rimozione o la correzione di dati rumorosi, incompleti o incoerenti. Tecniche come la deduplicazione dei dati, l'imputazione dei valori mancanti e la rimozione dei valori anomali possono essere applicate per migliorare la qualità dei dati di addestramento. Eliminando o filtrando i dati sospetti o anomali, è possibile ridurre l'impatto delle istanze avvelenate.
La validazione dei dati implica la verifica dell'integrità e della coerenza dei dati di training. Ciò può includere il controllo della coerenza del tipo di dati, la convalida dell'intervallo e le dipendenze tra campi. Definendo e applicando le regole di validazione dei dati, i dati anomali o incoerenti indicativi di avvelenamento possono essere identificati e segnalati per ulteriori indagini.
La provenienza dei dati e il tracciamento della discendenza implicano il mantenimento di un registro dell'origine, delle trasformazioni e dei movimenti dei dati in tutta la pipeline ML. Documentando le fonti dei dati, i passaggi di pre-elaborazione e qualsiasi modifica apportata, i professionisti possono risalire alle anomalie o ai pattern sospetti fino alla loro origine. Ciò aiuta a identificare potenziali punti di avvelenamento dei dati e facilita il processo di indagine e mitigazione.

17.4.4.2.3 Tecniche di Training Robusti

È possibile utilizzare tecniche di ottimizzazione robuste per modificare l'obiettivo del training per ridurre al minimo l'impatto di valori anomali o istanze avvelenate. Ciò può essere ottenuto utilizzando funzioni di perdita robuste meno sensibili ai valori estremi, come la "Huber loss" o la "modified Huber loss". Le tecniche di regolarizzazione, come la regolarizzazione L1 o L2, possono anche aiutare a ridurre la sensibilità del modello ai dati avvelenati, limitando la complessità del modello e prevenendo l'overfitting.
Le funzioni di "loss" [perdita] robuste sono progettate per essere meno sensibili ai valori anomali o ai dati rumorosi. Esempi includono la Huber loss modificata, la perdita di Tukey (Beaton e Tukey 1974) e la "trimmed mean loss". Queste funzioni di perdita riducono o ignorano il contributo delle istanze anomale durante il training, riducendo il loro impatto sul processo di apprendimento del modello. Le funzioni "obiettivo" robuste, come l'obiettivo

minimax o la “distributivamente robusto”, mirano a ottimizzare le prestazioni del modello negli scenari peggiori o in presenza di perturbazioni avversarie.
Le tecniche di “data augmentation” comportano la generazione di esempi di addestramento aggiuntivi applicando trasformazioni o perturbazioni casuali ai dati esistenti Figura 17.32. Ciò aiuta ad aumentare la diversità e la robustezza del set di dati di addestramento. Introducendo variazioni controllate nei dati, il modello diventa meno sensibile a pattern o artefatti specifici che possono essere presenti in istanze avvelenate. Le tecniche di randomizzazione, come il sottocampionamento casuale o l’aggregazione bootstrap, possono anche aiutare a ridurre l’impatto dei dati avvelenati addestrando più modelli su diversi sottoinsiemi di dati e combinando le loro previsioni.

Figura 17.32: Un’immagine del numero “3” nella forma originale e con applicati degli aumenti di base.

17.4.4.2.4 Approvvigionamento di Dati Sicuro e Affidabile

L’implementazione delle migliori pratiche di raccolta e cura dei dati può aiutare a mitigare il rischio di avvelenamento dei dati. Ciò include l’istituzione di protocolli di raccolta dati chiari, la verifica dell’autenticità e dell’affidabilità delle fonti dati e la conduzione di valutazioni regolari della qualità dei dati. L’approvvigionamento di dati da provider affidabili e rispettabili e il rispetto di pratiche di gestione dei dati sicure possono ridurre la probabilità di introdurre dati avvelenati nella pipeline di training.
Solidi meccanismi di governance dei dati e controllo degli accessi sono essenziali per prevenire modifiche non autorizzate o manomissioni dei dati di training. Ciò implica la definizione di ruoli e responsabilità chiari per l’accesso ai dati, l’implementazione di policy di controllo degli accessi basate sul principio del privilegio minimo e il monitoraggio e il logging delle attività di accesso ai dati. Limitando l’accesso ai dati di training e mantenendo un audit trail, è possibile rilevare e investigare potenziali tentativi di avvelenamento dei dati.
Rilevare e mitigare gli attacchi di avvelenamento dei dati richiede un approccio poliedrico che combini rilevamento delle anomalie, sanificazione dei dati, tecniche di training affidabili e pratiche di approvvigionamento dei dati sicure. Implementando queste misure, i professionisti del ML possono migliorare la resilienza dei loro modelli contro l’avvelenamento dei dati e garantire l’integrità e l’affidabilità dei dati di training. Tuttavia, è importante notare che l’avvelenamento dei dati è un’area di ricerca attiva e continuano a emergere nuovi vettori di attacco e meccanismi di difesa. Rimanere informati sugli ultimi sviluppi e adottare un approccio proattivo e adattivo alla sicurezza dei dati è fondamentale per mantenere la robustezza dei sistemi ML.

17.4.4.3 Distribution Shift

17.4.4.3.1 Rilevamento e Mitigazione dei “Distribution Shift”

Ricordiamo che i “distribution shift” [spostamenti di distribuzione] si verificano quando la distribuzione dei dati incontrata da un modello di machine learning (ML) durante l’implementazione differisce dalla distribuzione su cui è stato addestrato. Questi spostamenti possono avere un impatto significativo sulle prestazioni e sulla capacità di generalizzazione del modello, portando a previsioni non ottimali o errate. Rilevare e mitigare i

“distribution shift” è fondamentale per garantire la robustezza e l’affidabilità dei sistemi ML in scenari reali.

17.4.4.3.2 Tecniche di Rilevamento per i “Distribution Shift”

I test statistici possono essere utilizzati per confrontare le distribuzioni dei dati di training e di test per identificare differenze significative. Tecniche come il test di Kolmogorov-Smirnov o il test di Anderson-Darling misurano la discrepanza tra due distribuzioni e forniscono una valutazione quantitativa della presenza di un “distribution shift”. Applicando questi test alle funzionalità di input o alle previsioni del modello, i professionisti possono rilevare se esiste una differenza statisticamente significativa tra le distribuzioni di training e di test.
Le metriche di divergenza quantificano la dissimilarità tra due distribuzioni di probabilità. Le metriche di divergenza comunemente utilizzate includono la Divergenza Kullback-Leibler (KL) e la Divergenza Jensen-Shannon (JS). Calcolando la divergenza tra le distribuzioni dei dati di training e di test, i professionisti possono valutare l’entità dello “spostamento della distribuzione”. Valori di divergenza elevati indicano una differenza significativa tra le distribuzioni, suggerendo la presenza di uno spostamento della distribuzione.
Le tecniche di quantificazione dell’incertezza, come le reti neurali bayesiane o i metodi di ensemble, possono stimare l’incertezza associata alle previsioni del modello. Quando un modello viene applicato a dati da una distribuzione diversa, le sue previsioni potrebbero avere un’incertezza maggiore. Monitorando i livelli di incertezza, i professionisti possono rilevare gli spostamenti della distribuzione. Se l’incertezza supera costantemente una soglia predeterminata per i campioni di test, ciò suggerisce che il modello sta operando al di fuori della sua distribuzione addestrata.
Inoltre, i classificatori di dominio sono addestrati a distinguere tra diversi domini o distribuzioni. I professionisti possono rilevare gli spostamenti di distribuzione addestrando un classificatore a distinguere tra i domini di addestramento e di test. Se il classificatore di dominio raggiunge un’elevata accuratezza nel distinguere tra i due domini, indica una differenza significativa nelle distribuzioni sottostanti. Le prestazioni del classificatore di dominio servono come misura dello spostamento di distribuzione.

17.4.4.3.3 Tecniche di Mitigazione per i “Distribution Shift”

Il “transfer learning” [trasferimento dell’apprendimento.] sfrutta le conoscenze acquisite da un dominio per migliorare le prestazioni in un altro, come mostrato in Figura 17.33.
Utilizzando modelli pre-addestrati o trasferendo le feature apprese da un dominio di origine a un dominio di destinazione, il transfer learning può aiutare a mitigare l’impatto dei “distribution shift”. Il modello pre-addestrato può essere messo a punto su una piccola quantità di dati etichettati dal dominio target, consentendogli di adattarsi alla nuova distribuzione. Il transfer learning è particolarmente efficace quando i domini di origine e di destinazione condividono caratteristiche simili o quando i dati etichettati nel dominio di destinazione sono scarsi.

Figura 17.33: Trasferimento dell'apprendimento. Fonte: Bhavsar

L'apprendimento continuo, noto anche come apprendimento permanente, consente ai modelli ML di apprendere continuamente da nuove distribuzioni di dati, mantenendo al contempo le conoscenze delle distribuzioni precedenti. Tecniche come la "elastic weight consolidation (EWC)" (Kirkpatrick et al. 2017) o la "gradient episodic memory (GEM)" (Lopez-Paz e Ranzato 2017) consentono ai modelli di adattarsi alle distribuzioni di dati in evoluzione nel tempo. Queste tecniche mirano a bilanciare la plasticità del modello (capacità di apprendere da nuovi dati) con la stabilità del modello (mantenendo le conoscenze apprese in precedenza). Aggiornando gradualmente il modello con nuovi dati e mitigando l'oblio catastrofico, l'apprendimento continuo aiuta i modelli a rimanere robusti ai "distribution shift".

Le tecniche di aumento dei dati, come quelle viste in precedenza, comportano l'applicazione di trasformazioni o perturbazioni ai dati di training esistenti per aumentarne la diversità e migliorare la robustezza del modello ai "distribution shift". Introducendo variazioni nei dati, come rotazioni, traslazioni, ridimensionamenti o aggiunta di rumore, l'aumento dei dati aiuta il modello ad apprendere caratteristiche invarianti e a generalizzare meglio a distribuzioni mai viste. Il "data augmentation" può essere eseguito durante l'addestramento e l'inferenza per migliorare la capacità del modello di gestire i "distribution shift".

I metodi ensemble combinano più modelli per rendere le previsioni più robuste ai "distribution shift". Addestrando i modelli su diversi sottoinsiemi di dati, utilizzando algoritmi diversi o con diversi iperparametri, i metodi ensemble possono catturare diversi aspetti della distribuzione dei dati. Quando viene presentata una distribuzione "spostata", l'ensemble può sfruttare i punti di forza dei singoli modelli per fare previsioni più accurate e stabili. Tecniche come il bagging, il boosting o lo stacking possono creare ensemble efficaci.

Aggiornare regolarmente i modelli con nuovi dati dalla distribuzione target è fondamentale per mitigare l'impatto dei "distribution shift". Man mano che la distribuzione dei dati si evolve, i modelli dovrebbero essere riaddestrati o perfezionati sui dati disponibili più recenti per adattarsi ai pattern mutevoli. Il monitoraggio delle prestazioni del modello e delle caratteristiche dei dati può aiutare a rilevare quando è necessario un aggiornamento. Mantenendo aggiornati i modelli, i professionisti possono garantire che rimangano pertinenti e accurati di fronte ai distribution shift".

La valutazione dei modelli utilizzando metriche robuste meno sensibili ai "distribution shift" può fornire una valutazione più affidabile delle prestazioni del modello. Metriche come l'"area under the precision-recall curve (AUPRC)" [area sotto la curva di precisione-richiamo]

o il punteggio F1 sono più robuste allo squilibrio di classe e possono catturare meglio le prestazioni del modello su diverse distribuzioni. Inoltre, l'utilizzo di metriche di valutazione specifiche del dominio che si allineano con i risultati desiderati nel dominio target può fornire una misura più significativa dell'efficacia del modello.
Rilevare e mitigare i "distribution shift" è un processo continuo che richiede monitoraggio, adattamento e miglioramento continui. Utilizzando una combinazione di tecniche di rilevamento e strategie di mitigazione, i professionisti del ML possono identificare e affrontare in modo proattivo i "distribution shift", garantendo la robustezza e l'affidabilità dei loro modelli nelle distribuzioni del mondo reale. È importante notare che i "distribution shift" possono assumere varie forme e potrebbero richiedere approcci specifici del dominio a seconda della natura dei dati e dell'applicazione. Rimanere informati sulle ultime ricerche e sulle best practice nella gestione dei "distribution shift" è essenziale per creare sistemi ML resilienti.

17.5 Errori Software

17.5.1 Definizione e Caratteristiche

Gli errori software si riferiscono a difetti, errori o bug nei framework software runtime e nei componenti che supportano l'esecuzione e la distribuzione di modelli ML (Myllyaho et al. 2022). Questi guasti possono derivare da varie fonti, come errori di programmazione, difetti di progettazione o problemi di compatibilità (H. Zhang 2008), e possono avere implicazioni significative per le prestazioni, l'affidabilità e la sicurezza dei sistemi ML. Gli errori software nei framework ML presentano diverse caratteristiche chiave:

- **Diversità:** Gli errori software possono manifestarsi in forme diverse, che vanno da semplici errori di logica e sintassi a problemi più complessi come perdite di memoria, condizioni di "race" e problemi di integrazione. La varietà di tipi di errori aumenta la sfida di rilevarli e mitigarli in modo efficace.
- **Propagazione:** Nei sistemi ML, gli errori software possono propagarsi attraverso i vari layer e componenti del framework. Un errore in un modulo può innescare una cascata di errori o comportamenti imprevisti in altre parti del sistema, rendendo difficile individuare la causa principale e valutare l'impatto completo dell'errore.
- **Intermittenza:** Alcuni errori software possono presentare un comportamento intermittente, che si verifica sporadicamente o in condizioni specifiche. Questi errori possono essere particolarmente difficili da riprodurre e correggere, poiché possono manifestarsi in modo incoerente durante i test o il normale funzionamento.
- **Interazione con i modelli ML:** Gli errori software nei framework ML possono interagire con i modelli addestrati in modi sottili. Ad esempio, un errore nella pipeline di preelaborazione dei dati può introdurre rumore o distorsione negli input del modello, causando prestazioni degradate o previsioni errate. Analogamente, gli errori nel componente di servizio del modello possono causare incongruenze tra gli ambienti di training e inferenza.
- **Impatto sulle proprietà del sistema:** Gli errori software possono compromettere varie proprietà desiderabili dei sistemi ML, come prestazioni, scalabilità, affidabilità e sicurezza. Gli errori possono causare rallentamenti, crash, output errati o vulnerabilità che gli aggressori possono sfruttare.
- **Dipendenza da fattori esterni:** Il verificarsi e l'impatto degli errori software nei framework ML dipendono spesso da fattori esterni, come la scelta di hardware, sistema operativo, librerie e configurazioni. Problemi di compatibilità e mancate corrispondenze di versione possono introdurre errori difficili da anticipare e mitigare.

Comprendere le caratteristiche degli errori software nei framework ML è fondamentale per sviluppare strategie efficaci di prevenzione, rilevamento e mitigazione degli errori. Riconoscendo la diversità, la propagazione, l'intermittenza e l'impatto dei guasti software, i professionisti del ML possono progettare sistemi più robusti e affidabili, resilienti a questi problemi.

17.5.2 Meccanismi degli Errori Software nei Framework ML

I framework di apprendimento automatico, come TensorFlow, PyTorch e sci-kit-learn, forniscono potenti strumenti e astrazioni per la creazione e l'implementazione di modelli ML. Tuttavia, questi framework non sono immuni da errori software che possono influire sulle prestazioni, l'affidabilità e la correttezza dei sistemi ML. Esploriamo alcuni degli errori software comuni che possono verificarsi nei framework ML:

Memory Leak e Problemi di Gestione delle Risorse: Una gestione della memoria non corretta, come il mancato rilascio di memoria o la chiusura di handle di file, può portare a perdite di memoria e all'esaurimento delle risorse nel tempo. Questo problema è aggravato dall'utilizzo inefficiente della memoria, in cui la creazione di copie non necessarie di grandi tensori o il mancato sfruttamento di strutture dati efficienti in termini di memoria può causare un consumo eccessivo di memoria e degradare le prestazioni del sistema. Inoltre, la mancata gestione corretta della memoria GPU può causare errori di "out-of-memory" o un utilizzo non ottimale delle risorse GPU, aggravando ulteriormente il problema come mostrato in Figura 17.34.

```
RuntimeError: CUDA out of memory. Tried to allocate 200.00 MiB (GPU 0; 15.78 GiB total
capacity; 14.56 GiB already allocated; 38.44 MiB free; 14.80 GiB reserved in total by
PyTorch) If reserved memory is >> allocated memory try setting max_split_size_mb to avoid
fragmentation. See documentation for Memory Management and PYTORCH_CUDA_ALLOC_CONF
```

Figura 17.34: Esempio di problemi di memoria e utilizzo non ottimale della GPU

Problemi di Sincronizzazione e Concorrenza: Una sincronizzazione non corretta tra thread o processi può causare condizioni di "race", deadlock o comportamento incoerente nei sistemi ML multi-thread o distribuiti. Questo problema è spesso legato alla gestione impropria delle operazioni asincrone, come I/O non bloccante o caricamento dati parallelo, che può causare problemi di sincronizzazione e influire sulla correttezza della pipeline ML. Inoltre, un coordinamento e una comunicazione adeguati tra nodi distribuiti in un cluster possono causare coerenza o dati obsoleti durante l'addestramento o l'inferenza, compromettendo l'affidabilità del sistema ML.

Problemi di Compatibilità: Le discrepanze tra le versioni di framework, librerie o dipendenze ML possono introdurre problemi di compatibilità ed errori di runtime. L'aggiornamento o la modifica delle versioni delle librerie sottostanti senza testare a fondo l'impatto sul sistema ML può portare a comportamenti imprevisti o malfunzionamenti. Inoltre, le incongruenze tra gli ambienti di training e distribuzione, come differenze nell'hardware, nei sistemi operativi o nelle versioni dei pacchetti, possono causare problemi di compatibilità e influire sulla riproducibilità dei modelli ML, rendendo difficile garantire prestazioni coerenti sulle diverse piattaforme.

Instabilità Numerica ed Errori di Precisione: Una gestione inadeguata delle instabilità numeriche, come la divisione per zero, l'underflow o l'overflow, può portare a calcoli errati o problemi di convergenza durante l'addestramento. Questo problema è aggravato da errori di precisione o arrotondamento insufficienti, che possono accumularsi nel tempo e influire

sull'accuratezza dei modelli ML, specialmente nelle architetture di deep learning con molti livelli. Inoltre, un ridimensionamento o una normalizzazione impropri dei dati di input possono causare instabilità numeriche e influire sulla convergenza e sulle prestazioni degli algoritmi di ottimizzazione, con conseguenti prestazioni del modello non ottimali o inaffidabili.

Gestione degli Errori e delle Eccezioni Inadeguata: Una corretta gestione degli errori e delle eccezioni può impedire ai sistemi ML di bloccarsi o comportarsi in modo imprevisto quando si verificano condizioni eccezionali o input non validi. Non riuscire a catturare e gestire eccezioni specifiche o affidarsi alla gestione generica delle eccezioni può rendere difficile diagnosticare e recuperare gli errori in modo corretto, portando a instabilità del sistema e affidabilità ridotta. Inoltre, messaggi di errore incompleti o fuorvianti possono ostacolare la capacità di eseguire il debug e risolvere efficacemente gli errori software nei framework ML, prolungando il tempo necessario per identificare e risolvere i problemi.

17.5.3 Impatto sui Sistemi ML

Gli errori software nei framework di apprendimento automatico possono avere impatti significativi e di vasta portata sulle prestazioni, l'affidabilità e la sicurezza dei sistemi ML. Esploriamo i vari modi in cui gli errori software possono influenzare i sistemi ML:

Degrado delle Prestazioni e Rallentamenti del Sistema: Memory leak e gestione inefficiente delle risorse possono portare a un graduale degrado delle prestazioni nel tempo, poiché il sistema diventa sempre più vincolato dalla memoria e impiega più tempo nella garbage collection o nello swapping della memoria (Maas et al. 2024). Questo problema è aggravato da problemi di sincronizzazione e bug di concorrenza, che possono causare ritardi, riduzione della produttività e utilizzo non ottimale delle risorse di elaborazione, in particolare nei sistemi ML multi-thread o distribuiti. Inoltre, problemi di compatibilità o percorsi di codice inefficienti possono introdurre ulteriori overhead e rallentamenti, influenzando le prestazioni complessive del sistema ML.

Previsioni o Output Errati: Gli errori software nella pre-elaborazione dei dati, nell'ingegneria delle feature o nella valutazione del modello possono introdurre distorsioni, rumore o errori che si propagano attraverso la pipeline ML e che determinano previsioni o output errati. Nel tempo, instabilità numeriche, errori di precisione o problemi di arrotondamento possono accumularsi e portare a problemi di accuratezza o convergenza degradati nei modelli addestrati. Inoltre, gli errori nei componenti di servizio o inferenza del modello possono causare incongruenze tra gli output previsti e quelli effettivi, portando a previsioni errate o inaffidabili in produzione.

Problemi di Affidabilità e Stabilità: Gli errori software possono causare eccezioni senza precedenti, crash o terminazioni improvvise che possono compromettere l'affidabilità e la stabilità dei sistemi ML, specialmente negli ambienti di produzione. Gli errori intermittenti o sporadici possono essere difficili da riprodurre e diagnosticare, portando a un comportamento imprevedibile e a una ridotta fiducia negli output del sistema ML. Inoltre, errori nel checkpointing, nella serializzazione del modello o nella gestione dello stato possono causare perdite di dati o incongruenze, influenzando l'affidabilità e la recuperabilità del sistema ML.

Vulnerabilità di Sicurezza: Errori software, come buffer overflow, vulnerabilità di "injection" o controllo di accesso improprio, possono introdurre rischi per la sicurezza ed esporre il sistema ML a potenziali attacchi o accessi non autorizzati. Gli avversari possono sfruttare errori nelle fasi di pre-elaborazione o estrazione delle funzionalità per manipolare i dati di input e ingannare i modelli ML, portando a comportamenti errati o dannosi. Inoltre, una

protezione inadeguata dei dati sensibili, come le informazioni utente o i parametri riservati del modello, può portare a violazioni dei dati o violazioni della privacy (Q. Li et al. 2023).

Difficoltà nella Riproduzione e nel Debug: Gli errori software possono rendere difficile la riproduzione e il debug dei problemi nei sistemi ML, soprattutto quando gli errori sono intermittenti o dipendono da condizioni di runtime specifiche. Messaggi di errore incompleti o ambigui, uniti alla complessità dei framework e dei modelli ML, possono prolungare il processo di debug e ostacolare la capacità di identificare e correggere i guasti sottostanti. Inoltre, le incongruenze tra gli ambienti di sviluppo, test e produzione possono rendere difficile la riproduzione e la diagnosi dei guasti in contesti specifici.

Maggiori Costi di Sviluppo e Manutenzione I guasti software possono comportare maggiori costi di sviluppo e manutenzione, poiché i team dedicano più tempo e risorse al debug, alla correzione e alla validazione del sistema ML. La necessità di test estesi, monitoraggio e meccanismi di tolleranza agli errori per mitigare l'impatto degli errori software può aggiungere complessità e sovraccarico al processo di sviluppo ML. Patch, aggiornamenti e correzioni di bug frequenti per risolvere gli errori software possono interrompere il flusso di lavoro di sviluppo e richiedere sforzi aggiuntivi per garantire la stabilità e la compatibilità del sistema ML.
Comprendere il potenziale impatto degli errori software sui sistemi ML è fondamentale per dare priorità agli sforzi di test, implementare progetti di tolleranza agli errori e stabilire pratiche di monitoraggio e debug efficaci. Affrontando in modo proattivo gli errori software e le loro conseguenze, i professionisti ML possono creare sistemi ML più solidi, affidabili e sicuri che forniscono risultati accurati e affidabili.

17.5.4 Rilevamento e Mitigazione

Rilevare e mitigare i guasti software nei framework di apprendimento automatico è essenziale per garantire l'affidabilità, le prestazioni e la sicurezza dei sistemi ML. Esploriamo varie tecniche e approcci che possono essere impiegati per identificare e risolvere efficacemente i guasti software:

Test e Validazione Approfonditi: "Unit test" completi di singoli componenti e moduli possono verificarne la correttezza e identificare potenziali guasti nelle prime fasi dello sviluppo. I test di integrazione convalidano l'interazione e la compatibilità tra diversi componenti del framework ML, garantendo un'integrazione senza soluzione di continuità. I test sistematici di casi limite, condizioni al contorno e scenari eccezionali aiutano a scoprire guasti e vulnerabilità nascosti. Il "continuous testing" e i test di regressione come mostrato in

Figura 17.35 rilevano i guasti introdotti da modifiche al codice o aggiornamenti al framework ML.

Figura 17.35: Test di regressione automatizzati. Fonte: UTOR

Analisi Statica del Codice e Linting: L'utilizzo di strumenti di analisi statica del codice identifica automaticamente potenziali problemi di codifica, come errori di sintassi, variabili non definite o vulnerabilità di sicurezza. L'applicazione di standard di codifica e best practice tramite strumenti di "linting" mantiene la qualità del codice e riduce la probabilità di comuni errori di programmazione. L'esecuzione di revisioni regolari del codice consente l'ispezione manuale della base di codice, l'identificazione di potenziali errori e garantisce l'aderenza alle linee guida di codifica e ai principi di progettazione.

Monitoraggio e Logging in Fase di Esecuzione: L'implementazione di meccanismi di logging completi cattura informazioni rilevanti durante l'esecuzione, come dati di input, parametri del modello ed eventi di sistema. Il monitoraggio delle metriche delle prestazioni chiave, dell'utilizzo delle risorse e dei tassi di errore aiuta a rilevare anomalie, colli di bottiglia delle prestazioni o comportamenti imprevisti. L'impiego di controlli di asserzione in fase di esecuzione e invarianti, convalida le ipotesi e rileva violazioni delle condizioni previste durante l'esecuzione del programma. L'utilizzo di strumenti di profilazione consente di identificare colli di bottiglia nelle prestazioni, memory leak o percorsi di codice inefficienti che potrebbero indicare la presenza di errori software.

Design Pattern a Tolleranza di Errore: L'implementazione di meccanismi di gestione degli errori e delle eccezioni consente una gestione e un ripristino controllato da condizioni eccezionali o errori di runtime. L'impiego di meccanismi di ridondanza e failover, come sistemi di backup o calcoli ridondanti, garantisce la disponibilità e l'affidabilità del sistema ML in presenza di errori. La progettazione di architetture modulari e debolmente accoppiate riduce al minimo la propagazione e l'impatto dei guasti su diversi componenti del sistema ML. L'utilizzo di meccanismi di checkpointing e ripristino (Eisenman et al. 2022) consente al sistema di riprendere da uno stato stabile noto in caso di guasti o interruzioni.

Aggiornamenti e Patch Regolari: Rimanere aggiornati con le ultime versioni e patch dei framework, delle librerie e delle dipendenze ML offre vantaggi in termini di correzioni di bug, aggiornamenti di sicurezza e miglioramenti delle prestazioni. Il monitoraggio delle note di rilascio, degli avvisi di sicurezza e dei forum della community informa i professionisti su problemi noti, vulnerabilità o problemi di compatibilità nel framework ML. L'istituzione di un

processo sistematico per testare e convalidare aggiornamenti e patch prima di applicarli ai sistemi di produzione garantisce stabilità e compatibilità.

Containerizzazione e Isolamento: Sfruttando le tecnologie di containerizzazione, come Docker o Kubernetes, si incapsulano i componenti ML e le relative dipendenze in ambienti isolati. L'utilizzo della containerizzazione garantisce ambienti di runtime coerenti e riproducibili nelle fasi di sviluppo, test e produzione, riducendo la probabilità di problemi di compatibilità o errori specifici dell'ambiente. L'impiego di tecniche di isolamento, come ambienti virtuali o sandbox, impedisce che errori o vulnerabilità in un componente influiscano su altre parti del sistema ML.

Test Automatizzati e Continuous Integration/Continuous Deployment (CI/CD): Implementare framework e script di test automatizzati, eseguire suite di test complete e individuare gli errori nelle prime fasi dello sviluppo. L'integrazione di test automatizzati nella pipeline CI/CD, come mostrato in Figura 17.36, garantisce che le modifiche al codice siano testate a fondo prima di essere unite o distribuite in produzione. L'utilizzo di sistemi di monitoraggio continuo e di allerta automatizzati rilevano e notificano a sviluppatori e operatori potenziali guasti o anomalie in tempo reale.

Figura 17.36: Procedura di Continuous Integration/Continuous Deployment (CI/CD). Fonte: geeksforgeeks

L'adozione di un approccio proattivo e sistematico al rilevamento e alla mitigazione degli errori può migliorare significativamente la robustezza, l'affidabilità e la manutenibilità dei sistemi ML. Investendo in pratiche complete di test, monitoraggio e progettazione tollerante agli errori, le organizzazioni possono ridurre al minimo l'impatto degli errori software e garantire il regolare funzionamento dei loro sistemi ML negli ambienti di produzione.

Esercizio 17.4: Tolleranza agli Errori

Preparatevi a diventare supereroi che combattono gli errori dell'IA! I problemi software possono far deragliare i sistemi di apprendimento automatico, ma in questo Colab impareremo come renderli resilienti. Simuleremo errori software per vedere come l'IA può "rompersi", poi esploreremo tecniche per salvare i progressi del modello ML, come i checkpoint in un gioco. Vedremo come addestrare l'IA a riprendersi dopo un crash, assicurando che rimanga sulla buona strada. Questo è fondamentale per creare un'IA affidabile e degna di fiducia, soprattutto nelle applicazioni critiche. Quindi preparatevi perché questo Colab si collega direttamente al capitolo IA Robusta—passeremo dalla teoria alla risoluzione pratica dei problemi e creeremo sistemi di intelligenza artificiale in grado di gestire l'imprevisto!

Open in Colab

17.6 Strumenti e Framework

Data l'importanza di sviluppare sistemi di IA robusti, negli ultimi anni ricercatori e professionisti hanno sviluppato un'ampia gamma di strumenti e framework per comprendere

come i guasti hardware si manifestano e si propagano per avere un impatto sui sistemi ML. Questi strumenti e framework svolgono un ruolo cruciale nella valutazione della resilienza dei sistemi ML ai guasti hardware simulando vari scenari di guasto e analizzandone l'impatto sulle prestazioni del sistema. Ciò consente ai progettisti di identificare potenziali vulnerabilità e sviluppare strategie di mitigazione efficaci, creando in definitiva sistemi ML più robusti e affidabili in grado di funzionare in sicurezza nonostante i guasti hardware. Questa sezione fornisce una panoramica dei modelli di guasto ampiamente utilizzati nella letteratura e degli strumenti e framework sviluppati per valutare l'impatto di tali guasti sui sistemi ML.

17.6.1 Modelli di Guasto e Modelli di Errore

Come discusso in precedenza, i guasti hardware possono manifestarsi in vari modi, tra cui guasti transitori, permanenti e intermittenti. Oltre al tipo di guasto in esame, è importante anche *come* si manifesta il guasto. Ad esempio, l'errore si verifica in una cella di memoria o durante il calcolo di un'unità funzionale? L'impatto è su un singolo bit o su più bit? L'errore si propaga per tutto il percorso e ha un impatto sull'applicazione (causando un errore) o viene mascherato rapidamente ed è considerato benigno? Tutti questi dettagli hanno un impatto su ciò che è noto come *fault model* [modello di errore], che svolge un ruolo importante nella simulazione e nella misurazione di ciò che accade a un sistema quando si verifica un errore. Per studiare e comprendere efficacemente l'impatto degli errori hardware sui sistemi ML, è essenziale comprendere i concetti di "fault model" e "error model". Un "fault model" [guasto] descrive come si manifesta un errore hardware nel sistema, mentre un "error model" [modello di errore] rappresenta come l'errore si propaga e influisce sul comportamento del sistema.

I "fault model" possono essere categorizzati in base a varie caratteristiche:

- **Durata:** I guasti transitori si verificano brevemente e poi scompaiono, mentre quelli permanenti persistono indefinitamente. I guasti intermittenti si verificano sporadicamente e possono essere difficili da diagnosticare.
- **Posizione:** I guasti possono verificarsi in componenti hardware, come celle di memoria, unità funzionali o interconnessioni.
- **Granularità:** I guastipossono interessare un singolo bit (ad esempio, bitflip) o più bit (ad esempio, errori burst) all'interno di un componente hardware.

D'altro canto, gli "error model" descrivono come un guasto si propaga nel sistema e si manifesta come un errore. Un errore può causare la deviazione del sistema dal comportamento previsto, portando a risultati errati o persino a guasti del sistema. I modelli di errore possono essere definiti a diversi livelli di astrazione, da quello hardware (ad esempio, bitflip a livello di registro) al livello software (ad esempio, pesi o attivazioni corrotti in un modello ML).

Il "fault model" (o il modello di errore, in genere la terminologia più applicabile per comprendere la robustezza di un sistema ML) svolge un ruolo importante nella simulazione e nella misura di ciò che accade a un sistema quando si verifica un guasto. Il modello scelto informa le ipotesi fatte sul sistema in fase di studio. Ad esempio, un sistema incentrato su errori transitori a bit singolo (Sangchoolie, Pattabiraman, e Karlsson 2017) non sarebbe adatto a comprendere l'impatto di errori permanenti di flip multi-bit (Wilkening et al. 2014), poiché è progettato presupponendo un modello completamente diverso.

Inoltre, anche l'implementazione di un modello di errore è una considerazione importante, in particolare per quanto riguarda il punto in cui si dice che si verifichi un errore nello stack di elaborazione. Ad esempio, un modello di flip a bit singolo a livello di registro architetturale differisce da un modello di flip a bit singolo nel peso di un modello a livello di PyTorch. Sebbene entrambi mirino a un modello di errore simile, il primo verrebbe solitamente modellato in un simulatore architetturalmente accurato (come gem5 [binkert2011gem5]),

che cattura la propagazione dell'errore rispetto al secondo, concentrandosi sulla propagazione del valore attraverso un modello.
Ricerche recenti hanno dimostrato che alcune caratteristiche dei modelli di errore possono mostrare comportamenti simili a diversi livelli di astrazione (Sangchoolie, Pattabiraman, e Karlsson 2017) (Papadimitriou e Gizopoulos 2021). Ad esempio, gli errori a bit singolo sono generalmente più problematici degli errori a bit multiplo, indipendentemente dal fatto che siano modellati a livello hardware o software. Tuttavia, altre caratteristiche, come il mascheramento degli errori (Mohanram e Touba 2003) come mostrato in Figura 17.37, potrebbero non essere sempre catturate accuratamente dai modelli a livello software, poiché possono nascondere gli effetti di sistema sottostanti.

Figura 17.37: Esempio di mascheramento degli errori nei componenti microarchitettonici (Ko 2021)

Alcuni strumenti, come Fidelity (Y. He, Balaprakash, e Li 2020), mirano a colmare il divario tra modelli di errore a livello hardware e software mappando i pattern tra i due livelli di astrazione (E. Cheng et al. 2016). Ciò consente una modellazione più accurata dei guasti hardware negli strumenti basati su software, essenziale per lo sviluppo di sistemi ML robusti e affidabili. Gli strumenti a più basso livello in genere rappresentano caratteristiche di propagazione degli errori più accurate, ma devono essere più rapidi nella simulazione di molti errori a causa della natura complessa delle progettazioni dei sistemi hardware. D'altro canto, gli strumenti a più alto livello, come quelli implementati in framework ML come PyTorch o TensorFlow, di cui parleremo presto nelle sezioni successive, sono spesso più rapidi ed efficienti per valutare la robustezza dei sistemi ML.
Nelle sottosezioni seguenti, discuteremo vari metodi e strumenti di iniezione di guasti basati su hardware e software, evidenziandone le capacità, le limitazioni e i modelli di guasti ed errori che supportano.

17.6.2 Injection Hardware-based di Guasti

Uno strumento di "iniezione di errori" è uno strumento che consente all'utente di implementare un particolare modello di errore, come un singolo bit flip transitorio durante l'inferenza Figura 17.38. La maggior parte degli strumenti di iniezione di errori sono basati su software, poiché sono più rapidi per gli studi di robustezza ML. Tuttavia, i metodi di iniezione di guasti basati su hardware sono ancora importanti per radicare i modelli di errore ad alto livello, poiché sono considerati il modo più accurato per studiare l'impatto dei guasti sui sistemi ML manipolando direttamente l'hardware per introdurli. Questi metodi consentono ai ricercatori di osservare il comportamento del sistema in condizioni di guasti reali. In questa sezione vengono descritti in modo più dettagliato sia gli strumenti di iniezione di errori basati su software che quelli basati su hardware.

Figura 17.38: Gli errori hardware possono verificarsi per una serie di motivi e in momenti e/o posizioni diverse in un sistema, il che può essere esplorato quando si studia l'impatto degli errori basati sull'hardware sui sistemi (Ahmadilivani et al. 2024)

17.6.2.1 Metodi

Due dei metodi di iniezione di guasti basati su hardware più comuni sono quelli basati su FPGA e il test di radiazione o di fascio.

Iniezione di Guasti FPGA-based: I "Field-Programmable Gate Array (FPGA)" sono circuiti integrati riconfigurabili che possono essere programmati per implementare vari progetti hardware. Nel contesto dell'iniezione di guasti, gli FPGA offrono elevata precisione e accuratezza, poiché i ricercatori possono mirare a bit specifici o set di bit all'interno dell'hardware. Modificando la configurazione dell'FPGA, i guasti possono essere introdotti in posizioni e tempi specifici durante l'esecuzione di un modello ML. L'iniezione di guasti basata su FPGA consente un controllo dettagliato sul "fault model", consentendo ai ricercatori di studiare l'impatto di diversi tipi di guasti, come i flip di bit singoli o gli errori multi-bit. Questo livello di controllo rende l'iniezione di guasti basata su FPGA uno strumento prezioso per comprendere la resilienza dei sistemi ML ai guasti hardware.

Test di Radiazioni o Fasci: Il test di radiazioni o fasci (Velazco, Foucard, e Peronnard 2010) comporta l'esposizione dell'hardware che esegue un modello ML a particelle ad alta energia, come protoni o neutroni, come illustrato in Figura 17.39. Queste particelle possono causare bitflip o altri tipi di guasti nell'hardware, imitando gli effetti di quelli indotti dalle radiazioni nel mondo reale. Il test di fasci è ampiamente considerato un metodo altamente accurato per misurare il tasso di errore indotto da impatti di particelle su un'applicazione in esecuzione. Fornisce una rappresentazione realistica dei guasti in ambienti reali, in particolare in applicazioni esposte ad alti livelli di radiazioni, come sistemi spaziali o esperimenti di fisica delle particelle. Tuttavia, a differenza dell'iniezione di guasti basata su FPGA, il test di fasci potrebbe essere più preciso nel puntare a bit o componenti specifici all'interno dell'hardware, poiché potrebbe essere difficile puntare il fascio di particelle a un bit

particolare nell'hardware. Nonostante sia piuttosto costoso dal punto di vista della ricerca, il test del fascio è una pratica industriale molto apprezzata per l'affidabilità.

Figura 17.39: Configurazione del test di radiazione per componenti semiconduttori (Lee et al. 2022) Fonte: JD Instrument

17.6.2.2 Limitazioni

Nonostante la loro elevata accuratezza, i metodi di iniezione di guasti basati su hardware presentano diverse limitazioni che possono ostacolarne l'adozione diffusa:

Costo: L'iniezione di guasti e il test del fascio basati su FPGA richiedono hardware e strutture specializzate, la cui configurazione e manutenzione possono essere costose. Il costo di questi metodi può rappresentare un ostacolo significativo per ricercatori e organizzazioni con risorse limitate.

Scalabilità: I metodi basati su hardware sono generalmente più lenti e meno scalabili rispetto ai metodi basati su software. L'iniezione di guasti e la raccolta di dati sull'hardware possono richiedere tempo, limitando il numero di esperimenti eseguiti in un determinato lasso di tempo. Ciò può essere particolarmente impegnativo quando si studia la resilienza di sistemi ML su larga scala o si conducono analisi statistiche che richiedono molti esperimenti di iniezione di guasti.

Flessibilità: I metodi basati su hardware potrebbero non essere flessibili quanto quelli basati su software in termini di gamma di modelli di guasto e modelli di errore che possono supportare. Modificare la configurazione hardware o l'impostazione sperimentale per adattarsi a diversi modelli di errore può essere più impegnativo e richiedere più tempo rispetto ai metodi basati su software.
Nonostante queste limitazioni, i metodi di iniezione di errori basati su hardware rimangono strumenti essenziali per convalidare l'accuratezza dei metodi basati su software e per studiare l'impatto degli errori sui sistemi ML in contesti realistici. Combinando metodi basati su hardware e basati su software, i ricercatori possono acquisire una comprensione più

completa della resilienza dei sistemi ML ai guasti hardware e sviluppare strategie di mitigazione efficaci.

17.6.3 Strumenti di Injection di Guasti Software-based

Con il rapido sviluppo di framework ML negli ultimi anni, gli strumenti di iniezione di guasti basati su software hanno guadagnato popolarità nello studio della resilienza dei sistemi ML ai guasti hardware. Questi strumenti simulano gli effetti dei guasti hardware modificando la rappresentazione software del modello ML o il grafo computazionale sottostante. L'ascesa di framework ML come TensorFlow, PyTorch e Keras ha facilitato lo sviluppo di strumenti di iniezione di guasti che sono strettamente integrati con questi framework, rendendo più facile per i ricercatori condurre esperimenti di iniezione di guasti e analizzare i risultati.

17.6.3.0.1 Vantaggi e Compromessi

Gli strumenti di iniezione di guasti basati su software offrono diversi vantaggi rispetto a quelli basati su hardware:

Velocità: Gli strumenti basati su software sono generalmente più rapidi dei metodi basati su hardware, poiché non richiedono la modifica dell'hardware fisico o la configurazione di apparecchiature specializzate. Ciò consente ai ricercatori di condurre più esperimenti di iniezione di guasti in tempi più brevi, consentendo analisi più complete della resilienza dei sistemi ML.

Flessibilità: Gli strumenti basati su software sono più flessibili di quelli basati su hardware in termini di gamma di modelli di guasti ed errori che possono supportare. I ricercatori possono facilmente modificare l'implementazione software dello strumento di iniezione di guasti per adattarsi a diversi modelli di guasti o per indirizzare componenti specifici del sistema ML.

Accessibilità: Gli strumenti basati su software sono più accessibili dei metodi basati su hardware, poiché non richiedono hardware o strutture specializzate. Ciò semplifica per ricercatori e professionisti condurre esperimenti di iniezione di guasti e studiare la resilienza dei sistemi ML, anche con risorse limitate.

17.6.3.0.2 Limitazioni

Gli strumenti di iniezione di guasti basati su software presentano anche alcune limitazioni rispetto ai metodi basati su hardware:

Precisione: Gli strumenti basati su software potrebbero non sempre catturano l'intera gamma di effetti che i guasti hardware possono avere sul sistema. Poiché questi strumenti operano a un livello di astrazione più elevato, potrebbero dover recuperare alcune delle interazioni hardware di basso livello e dei meccanismi di propagazione degli errori che possono influire sul comportamento del sistema ML.

Fedeltà: Gli strumenti basati su software potrebbero fornire un livello di fedeltà diverso rispetto ai metodi basati su hardware in termini di rappresentazione delle condizioni di guasto del mondo reale. L'accuratezza dei risultati ottenuti dagli esperimenti di iniezione di guasti basati su software potrebbe dipendere da quanto il modello software si avvicini al comportamento hardware effettivo.

17.6.3.0.3 Tipi di Strumenti di Iniezione di Guasti

Gli strumenti di iniezione di guasti basati su software possono essere categorizzati in base ai loro framework di destinazione o casi d'uso. Qui, discuteremo alcuni degli strumenti più popolari in ciascuna categoria:

Ares (Reagen et al. 2018), uno strumento di iniezione di guasti inizialmente sviluppato per il framework Keras nel 2018, è emerso come uno dei primi strumenti per studiare l'impatto dei guasti hardware sulle reti deep neural network (DNN) nel contesto della crescente popolarità

dei framework ML a metà-fine anni 2010. Lo strumento è stato convalidato rispetto a un acceleratore DNN implementato in silicio, dimostrando la sua efficacia nella modellazione dei guasti hardware. Ares fornisce uno studio completo sull'impatto dei guasti hardware sia nei pesi che nei valori di attivazione, caratterizzando gli effetti dei flip di bit singoli e dei bit-error rate (BER) sulle strutture hardware. Successivamente, il framework Ares è stato esteso per supportare l'ecosistema PyTorch, consentendo ai ricercatori di investigare i guasti hardware in un contesto più moderno e ampliando ulteriormente la sua utilità sul campo. PyTorchFI (Mahmoud et al. 2020), uno strumento di iniezione di guasti progettato specificamente per il framework PyTorch, è stato sviluppato nel 2020 in collaborazione con Nvidia Research. Consente l'iniezione di guasti nei pesi, nelle attivazioni e nei gradienti dei modelli PyTorch, supportando un'ampia gamma di modelli di guasti. Sfruttando le capacità di accelerazione GPU di PyTorch, PyTorchFI fornisce un'implementazione rapida ed efficiente per condurre esperimenti di iniezione di guasti su sistemi ML su larga scala, come mostrato in Figura 17.40.

Figura 17.40: I bitflip hardware nei carichi di lavoro ML possono causare oggetti fantasma e classificazioni errate, che possono essere erroneamente utilizzati a valle da sistemi più grandi, come nella guida autonoma. Quella mostrata sopra è una versione corretta e difettosa della stessa immagine che utilizza il framework di iniezione PyTorchFI.

La velocità e la facilità d'uso dello strumento hanno portato a un'adozione diffusa nella comunità, con conseguenti molteplici progetti guidati dagli sviluppatori, come PyTorchALFI di Intel xColabs, che si concentra sulla sicurezza negli ambienti automobilistici. Gli strumenti successivi incentrati su PyTorch per l'iniezione di guasti includono Dr. DNA di Meta (Ma et al. 2024) (che facilita ulteriormente il modello di programmazione "Pythonico" per facilità d'uso) e il framework GoldenEye (Mahmoud et al. 2022), che incorpora nuovi tipi di dati numerici (come AdaptivFloat (Tambe et al. 2020) e BlockFloat nel contesto di bit flip hardware. TensorFI (Zitao Chen et al. 2020), o TensorFlow Fault Injector, è uno strumento di iniezione di guasti sviluppato specificamente per il framework TensorFlow. Analogo ad Ares e PyTorchFI, TensorFI è considerato lo strumento all'avanguardia per gli studi di robustezza ML nell'ecosistema TensorFlow. Consente ai ricercatori di iniettare guasti nel grafo computazionale di Modelli TensorFlow e studia il loro impatto sulle prestazioni del modello, supportando un'ampia gamma di modelli di errore. Uno dei principali vantaggi di TensorFI è la sua capacità di valutare la resilienza di vari modelli ML, non solo DNN. Ulteriori progressi, come BinFi (Zitao Chen et al. 2019), forniscono un meccanismo per accelerare gli esperimenti di iniezione di errori concentrandosi sui bit "importanti" nel sistema, accelerando il processo di analisi della robustezza ML e dando priorità ai componenti critici di un modello.

NVBitFI (T. Tsai et al. 2021), uno strumento di iniezione di errori generico sviluppato da Nvidia per le sue piattaforme GPU, opera a un più basso livello rispetto a strumenti specifici del framework come Ares, PyTorchFI e TensorFlow. Mentre questi strumenti si concentrano su varie piattaforme di deep learning per implementare ed eseguire analisi di robustezza, NVBitFI mira al codice di assemblaggio hardware sottostante per l'iniezione di guasti. Ciò consente ai ricercatori di iniettare guasti in qualsiasi applicazione in esecuzione su GPU Nvidia, rendendolo uno strumento versatile per studiare la resilienza dei sistemi ML e di altre applicazioni accelerate da GPU. Consentendo agli utenti di iniettare errori a livello di architettura, NVBitFI fornisce un modello di guasto più generico che non è limitato ai soli modelli ML. Poiché i sistemi GPU di Nvidia sono comunemente utilizzati in molti sistemi basati su ML, NVBitFI è uno strumento prezioso per un'analisi completa dell'iniezione di guasti in varie applicazioni.

17.6.3.0.3.1 Esempi specifici di dominio

Sono stati sviluppati strumenti di iniezione di guasti specifici per dominio per affrontare le sfide e i requisiti unici di vari domini applicativi ML, come veicoli autonomi e robotica. Questa sezione evidenzia tre strumenti di iniezione di guasti specifici per dominio: DriveFI e PyTorchALFI per veicoli autonomi e MAVFI per "uncrewed aerial vehicles (UAV)" [veicoli aerei senza equipaggio]. Questi strumenti consentono ai ricercatori di iniettare guasti hardware nei sottosistemi di percezione, controllo e altri sistemi complessi, consentendo loro di studiare l'impatto dei guasti sulle prestazioni e sulla sicurezza del sistema. Lo sviluppo di questi strumenti di iniezione di guasti basati su software ha notevolmente ampliato le capacità della comunità ML di sviluppare sistemi più robusti e affidabili in grado di funzionare in modo sicuro ed efficace in presenza di guasti hardware.

DriveFI (S. Jha et al. 2019) è uno strumento di iniezione di guasti progettato per veicoli autonomi. Consente l'iniezione di guasti hardware nelle pipeline di percezione e controllo dei sistemi di veicoli autonomi, consentendo ai ricercatori di studiare l'impatto di questi guasti sulle prestazioni e sulla sicurezza del sistema. DriveFI è stato integrato con piattaforme di guida autonoma standard del settore, come Nvidia DriveAV e Baidu Apollo, rendendolo uno strumento prezioso per valutare la resilienza dei sistemi di veicoli autonomi.

PyTorchALFI (Gräfe et al. 2023) è un'estensione di PyTorchFI sviluppata da Intel xColabs per il dominio dei veicoli autonomi. Si basa sulle capacità di inserimento di guasti di PyTorchFI. Aggiunge funzionalità specificamente studiate per valutare la resilienza dei sistemi di veicoli autonomi, come la capacità di inserire guasti nei dati della telecamera e del sensore LiDAR.

MAVFI (Hsiao et al. 2023) è uno strumento di inserimento di guasti progettato per il dominio della robotica, in particolare per i veicoli aerei senza equipaggio (UAV). MAVFI è basato sul framework Robot Operating System (ROS) e consente ai ricercatori di inserire guasti nei vari componenti di un sistema UAV, come sensori, attuatori e algoritmi di controllo. Valutando l'impatto di questi guasti sulle prestazioni e sulla stabilità del UAV, i ricercatori possono sviluppare sistemi UAV più resilienti e tolleranti ai guasti.

Lo sviluppo di strumenti di iniezione di guasti basati su software ha notevolmente ampliato le capacità di ricercatori e professionisti di studiare la resilienza dei sistemi ML ai guasti hardware. Sfruttando la velocità, la flessibilità e l'accessibilità di questi strumenti, la comunità ML può sviluppare sistemi più robusti e affidabili in grado di funzionare in modo sicuro ed efficace in presenza di guasti hardware.

17.6.4 Colmare il Divario tra Modelli di Errore Hardware e Software

Sebbene gli strumenti di iniezione di guasti basati su software offrano molti vantaggi in termini di velocità, flessibilità e accessibilità, potrebbero non sempre catturare accuratamente l'intera gamma di effetti che i guasti hardware possono avere sul sistema. Questo perché gli strumenti basati su software operano a un livello di astrazione più alto

rispetto ai metodi basati su hardware e potrebbero non rilevare alcune delle interazioni hardware di basso livello e dei meccanismi di propagazione degli errori che possono influire sul comportamento del sistema ML.
Come illustra Bolchini et al. (2023) nel suo lavoro, gli errori hardware possono manifestarsi in complessi pattern di distribuzione spaziale che sono difficili da replicare completamente con la sola iniezione di guasti basata su software. Identificano quattro pattern distinti: (a) singolo punto, in cui il guasto corrompe un singolo valore in una feature map; (b) stessa riga, in cui il guasto corrompe una riga parziale o intera in una singola feature map; (c) bullet wake, in cui il guasto corrompe la stessa posizione su più feature map; e (d) shatter glass, che combina gli effetti dei pattern della stessa riga e bullet wake, come mostrato in Figura 17.41. Questi intricati meccanismi di propagazione degli errori evidenziano la necessità di tecniche di iniezione di guasti consapevoli dell'hardware per valutare accuratamente la resilienza dei sistemi ML.

Figure 9. Spatial distribution patterns (erroneous values are colored in red). (a) *Single point*: the fault causes the corruption of a single value of a single feature map. (b) *Same row*: the fault causes the total or partial corruption of a row in a single feature map. (c) *Bullet wake*: the fault corrupts the same location in all or multiple feature maps. (d) *Shatter glass*: the fault causes the combination of the effects of *same row* and *bullet wake* patterns.

Figura 17.41: Gli errori hardware possono manifestarsi in modi diversi a livello software, come classificato da Bolchini et al. (Bolchini et al. 2023)

I ricercatori hanno sviluppato strumenti per affrontare questo problema colmando il divario tra modelli di errore hardware di basso livello e modelli di errore software di livello superiore. Uno di questi strumenti è Fidelity, progettato per mappare i pattern tra guasti a livello hardware e le loro manifestazioni a livello software.

17.6.4.1 Fidelity: Colmare il Gap

Fidelity (Y. He, Balaprakash, e Li 2020) è uno strumento per modellare accuratamente i guasti hardware negli esperimenti di iniezione di guasti basati su software. Ciò avviene studiando attentamente la relazione tra i guasti a livello hardware e il loro impatto sulla rappresentazione software del sistema ML.
Le intuizioni chiave alla base di Fidelity sono:

- **Propagazione dei Guasti:** Fidelity modella il modo in cui gli errori si propagano attraverso l'hardware e si manifestano come errori nello stato del sistema visibili al software. Comprendendo questi pattern di propagazione, Fidelity può simulare con maggiore accuratezza gli effetti dei guasti hardware negli esperimenti basati sul software.

- **Equivalenza dei Guasti:** Fidelity identifica classi equivalenti di guasti hardware che producono errori simili a livello software. Ciò consente ai ricercatori di progettare modelli di guasti basati sul software che siano rappresentativi dei guasti hardware sottostanti senza la necessità di modellare ogni possibile guasto hardware singolarmente.
- **Approccio a Strati:** Fidelity impiega un approccio a strati alla modellazione dei guasti, in cui gli effetti dei guasti hardware vengono propagati attraverso più livelli di astrazione, dall'hardware al livello software. Questo approccio garantisce che i modelli di guasti basati sul software siano basati sul comportamento effettivo dell'hardware.

Incorporando queste informazioni, Fidelity consente agli strumenti di iniezione di guasti basati su software di catturare con precisione gli effetti dei guasti hardware sui sistemi ML. Ciò è particolarmente importante per le applicazioni critiche per la sicurezza, in cui la resilienza del sistema ai guasti hardware è fondamentale.

17.6.4.2 L'Importanza di Catturare il Vero Comportamento Hardware

Catturare il vero comportamento hardware negli strumenti di iniezione di guasti basati su software è fondamentale per diversi motivi:

- **Precisione:** Modellando con precisione gli effetti dei guasti hardware, gli strumenti basati su software possono fornire informazioni più affidabili sulla resilienza dei sistemi ML. Ciò è essenziale per progettare e convalidare sistemi tolleranti ai guasti che possono funzionare in modo sicuro ed efficace in presenza di guasti hardware.
- **Riproducibilità:** Quando gli strumenti basati su software catturano con precisione il comportamento hardware, gli esperimenti di iniezione di guasti diventano più riproducibili su diverse piattaforme e ambienti. Ciò è importante per lo studio scientifico della resilienza del sistema ML, poiché consente ai ricercatori di confrontare e convalidare i risultati su diversi studi e implementazioni.
- **Efficienza:** Gli strumenti basati su software che catturano il vero comportamento dell'hardware possono essere più efficienti nei loro esperimenti di iniezione di guasti concentrandosi sui modelli di guasti più rappresentativi e impattanti. Ciò consente ai ricercatori di coprire una gamma più ampia di scenari di guasti e configurazioni di sistema con risorse computazionali limitate.
- **Strategie di Mitigazione:** Comprendere come i guasti hardware si manifestano a livello software è fondamentale per sviluppare strategie di mitigazione efficaci. Catturando con precisione il comportamento dell'hardware, gli strumenti di iniezione di guasti basati su software possono aiutare i ricercatori a identificare i componenti più vulnerabili del sistema ML e progettare tecniche di rafforzamento mirate per migliorare la resilienza.

Strumenti come Fidelity sono essenziali per far progredire lo stato dell'arte nella ricerca sulla resilienza del sistema ML. Questi strumenti consentono ai ricercatori di condurre esperimenti di iniezione di guasti più accurati, riproducibili ed efficienti colmando il divario tra modelli di errore hardware e software. Man mano che la complessità e la criticità dei sistemi ML continuano a crescere, l'importanza di catturare il vero comportamento hardware negli strumenti di iniezione di guasti basati su software diventerà sempre più evidente.

La ricerca in corso in quest'area cerca di perfezionare la mappatura tra modelli di errore hardware e software e di sviluppare nuove tecniche per simulare in modo efficiente i guasti hardware negli esperimenti basati su software. Man mano che questi strumenti maturano, forniranno alla comunità ML mezzi sempre più potenti e accessibili per studiare e migliorare la resilienza dei sistemi ML ai guasti hardware.

17.7 Conclusione

Sviluppare un'IA solida e resiliente è fondamentale man mano che i sistemi di apprendimento automatico diventano sempre più integrati in applicazioni critiche per la sicurezza e in ambienti reali. Questo capitolo ha esplorato le principali sfide alla robustezza dell'IA derivanti da guasti hardware, attacchi dannosi, cambiamenti di distribuzione e bug software.

Alcune delle conclusioni principali includono quanto segue:

- **Guasti Hardware:** Guasti transitori, permanenti e intermittenti nei componenti hardware possono corrompere i calcoli e degradare le prestazioni dei modelli di apprendimento automatico se non vengono rilevati e mitigati correttamente. Tecniche come ridondanza, correzione degli errori e progetti fault-tolerant svolgono un ruolo cruciale nella creazione di sistemi ML resilienti in grado di resistere ai guasti hardware.
- **Robustezza del Modello:** Gli attori malintenzionati possono sfruttare le vulnerabilità nei modelli ML tramite attacchi avversari e avvelenamento dei dati, mirando a indurre classificazioni errate mirate, distorcere il comportamento appreso del modello o compromettere l'integrità e l'affidabilità del sistema. Inoltre, possono verificarsi "distribution shift" quando la distribuzione dei dati riscontrata durante l'implementazione differisce da quella osservata durante il training, con conseguente degrado delle prestazioni. L'implementazione di misure difensive, tra cui training avversario, rilevamento delle anomalie, architetture di modelli robuste e tecniche come adattamento del dominio, apprendimento per trasferimento e apprendimento continuo, è essenziale per proteggersi da queste sfide e garantire l'affidabilità e la generalizzazione del modello in ambienti dinamici.
- **Errori Software:** Gli errori nei framework ML, nelle librerie e negli stack software possono propagarsi, degradare le prestazioni e introdurre vulnerabilità di sicurezza. Test rigorosi, monitoraggio del runtime e adozione di "design pattern" tolleranti agli errori sono essenziali per la creazione di un'infrastruttura software robusta che supporti sistemi ML affidabili.

Poiché i sistemi ML affrontano attività sempre più complesse con conseguenze nel mondo reale, dare priorità alla resilienza diventa fondamentale. Gli strumenti e i framework discussi in questo capitolo, tra cui tecniche di "fault injection" [iniezione di guasti], metodi di analisi degli errori e framework di valutazione della robustezza, forniscono ai professionisti i mezzi per testare a fondo e rafforzare i propri sistemi ML contro varie modalità di errore e condizioni avverse.

Andando avanti, la resilienza deve essere un obiettivo centrale durante l'intero ciclo di vita dello sviluppo dell'IA, dalla raccolta dei dati e dall'addestramento del modello all'implementazione e al monitoraggio. Affrontando in modo proattivo le molteplici sfide alla robustezza, possiamo sviluppare sistemi di apprendimento automatico affidabili e sicuri, in grado di affrontare le complessità e le incertezze degli ambienti del mondo reale.

La ricerca futura sul ML robusto dovrebbe continuare a far progredire le tecniche per rilevare e mitigare guasti, attacchi e "shift" delle distribuzioni. Inoltre, esplorare nuovi paradigmi per lo sviluppo di architetture IA intrinsecamente resilienti, come sistemi auto-riparanti o meccanismi a prova di errore, sarà fondamentale per spingere i confini della robustezza dell'IA. Dando priorità alla resilienza e investendo nello sviluppo di sistemi di IA robusti, possiamo liberare il pieno potenziale delle tecnologie di apprendimento automatico, garantendone al contempo un'implementazione sicura, affidabile e responsabile in applicazioni del mondo reale. Mentre l'IA continua a plasmare il nostro futuro, la creazione di sistemi resilienti in grado di resistere alle sfide del mondo reale sarà un fattore determinante per il successo e l'impatto sociale di questa tecnologia trasformativa.

17.8 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Lavoriamo continuamente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo sia gli studenti che gli insegnanti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- *Prossimamente*.

Video

- *Prossimamente*.

Esercizi

Per rafforzare i concetti trattati in questo capitolo, abbiamo curato una serie di esercizi che sfidano gli studenti ad applicare le proprie conoscenze e ad approfondire la propria comprensione.

- Esercizio 17.1
- Esercizio 17.2
- Esercizio 17.3
- Esercizio 17.4

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente*.

18. IA Generativa

Prossimamente!

Immaginate un capitolo che si scrive da solo e si adatta alla propria curiosità, generando nuove intuizioni mentre si legge. Stiamo lavorando a qualcosa di straordinario!

Questo capitolo trasformerà il modo di leggere e imparare, generando dinamicamente contenuti man mano che si procede. Mentre perfezioniamo questa nuova entusiasmante funzionalità, speriamo che gli utenti si preparino per un'esperienza educativa dinamica e unica come. Segnate sul calendario la grande rivelazione e aggiungete questa pagina a quelle preferite.

*Il futuro dell'**apprendimento generativo** è qui! — Vijay Janapa Reddi*

19. AI for Good

DALL·E 3 Prompt: Illustrazione del pianeta Terra avvolto in reti neurali scintillanti, con diversi esseri umani e robot IA che lavorano insieme a vari progetti come piantare alberi, pulire gli oceani e sviluppare soluzioni energetiche sostenibili. L'atmosfera positiva e piena di speranza rappresenta uno sforzo congiunto per creare un futuro migliore.

Allineando il progresso dell'IA con i valori, gli obiettivi e l'etica umani, l'obiettivo finale dei sistemi ML (a qualsiasi scala) è quello di essere una tecnologia che rifletta i principi e le aspirazioni umane. Le iniziative nell'ambito di "AI for Good" promuovono lo sviluppo dell'IA per affrontare gli "UN Sustainable Development Goals (SDG)" [Obiettivi di sviluppo sostenibile delle Nazioni Unite] utilizzando tecnologie di IA embedded, ampliando l'accesso all'istruzione sull'IA, tra le altre cose. Sebbene sia ormai chiaro che l'IA sarà una parte fondamentale del progresso verso gli SDG, la sua adozione e il suo impatto sono limitati dall'enorme consumo di energia, dai forti requisiti di connettività e dagli elevati costi delle distribuzioni basate su cloud. TinyML può aggirare molti di questi problemi consentendo ai modelli ML di funzionare su microcontrollori a basso costo e basso consumo.

> Il movimento "AI for Good" è fondamentale per coltivare un futuro in cui una società potenziata dall'IA sia più giusta, sostenibile e prospera per tutta l'umanità.

Obiettivi dell'Apprendimento

- Comprendere come TinyML possa contribuire a promuovere gli "Obiettivi di Sviluppo Sostenibile delle Nazioni Unite" in materia di salute, agricoltura, istruzione e ambiente.
- Riconoscere la versatilità di TinyML per abilitare soluzioni localizzate e a basso costo, su misura per le esigenze della comunità.
- Considerare le sfide dell'adozione di TinyML a livello globale, come training limitato, vincoli di dati, accessibilità e barriere culturali.
- Apprezzare l'importanza di approcci collaborativi ed etici per sviluppare e distribuire TinyML per servire al meglio i contesti locali.
- Riconoscere il potenziale di TinyML, se implementato in modo responsabile, per promuovere l'equità e dare potere alle popolazioni svantaggiate in tutto il mondo.

19.1 Introduzione

Per darci un quadro attorno al quale riflettere sull'intelligenza artificiale per il bene sociale, seguiremo gli "Obiettivi di Sviluppo Sostenibile delle Nazioni Unite (SDG)". Gli SDG delle Nazioni Unite sono una raccolta di 17 obiettivi globali, mostrati in Figura 19.1, adottati dalle Nazioni Unite nel 2015 come parte dell'Agenda 2030 per lo Sviluppo Sostenibile. Gli SDG affrontano le sfide globali relative a povertà, disuguaglianza, cambiamenti climatici, degrado ambientale, prosperità, pace e giustizia.

Figura 19.1: United Nations Sustainable Development Goals (SDG). Fonte: United Nations.

Ciò che rende speciali gli SDG è che sono una raccolta di obiettivi interconnessi progettati per fungere da "modello condiviso per la pace e la prosperità per le persone e il pianeta, ora e in futuro". Gli SDG enfatizzano gli aspetti ambientali, sociali ed economici interconnessi dello sviluppo sostenibile, ponendo la sostenibilità al centro.

Uno studio recente (Vinuesa et al. 2020) evidenzia l'influenza dell'IA su tutti gli aspetti dello sviluppo sostenibile, in particolare sui 17 "Sustainable Development Goals (SDG)" e 169 target definiti a livello internazionale nell'Agenda 2030 per lo Sviluppo Sostenibile. Lo studio mostra che l'IA può fungere da abilitatore per 134 target attraverso miglioramenti tecnologici, ma evidenzia anche le sfide dell'IA su alcuni target. Lo studio mostra che l'IA può avvantaggiare 67 target quando si considerano l'IA e i risultati sociali. Tuttavia, mette anche in guardia sui problemi relativi all'implementazione dell'IA in paesi con valori culturali e ricchezza diversi.

Nel contesto del nostro libro, TinyML potrebbe aiutare a promuovere almeno alcuni di questi obiettivi SDG.

- **Obiettivo 1 - Nessuna Povertà:** TinyML potrebbe aiutare a fornire soluzioni a basso costo per il monitoraggio delle colture per migliorare le rese agricole nei paesi in via di sviluppo.
- **Obiettivo 2 - Zero Fame:** TinyML potrebbe consentire un monitoraggio localizzato e preciso della salute delle colture e il rilevamento delle malattie per ridurre le perdite di raccolto.

- **Obiettivo 3 - Buona Salute e Benessere:** TinyML potrebbe aiutare a abilitare strumenti di diagnosi medica a basso costo per la diagnosi precoce e la prevenzione delle malattie nelle aree remote.
- **Obiettivo 6 - Acqua Pulita e Servizi igienici:** TinyML potrebbe monitorare la qualità dell'acqua e rilevare i contaminanti per garantire l'accesso all'acqua potabile.
- **Obiettivo 7 - Energia Pulita e Accessibile:** TinyML potrebbe ottimizzare il consumo di energia e consentire la manutenzione predittiva per le infrastrutture di energia rinnovabile.
- **Obiettivo 11 - Città e Comunità Sostenibili:** TinyML potrebbe consentire una gestione intelligente del traffico, il monitoraggio della qualità dell'aria e una gestione ottimizzata delle risorse nelle città intelligenti.
- **Obiettivo 13 - Azione per il Clima:** TinyML potrebbe monitorare la deforestazione e tracciare gli sforzi di riforestazione. Potrebbe anche aiutare a prevedere eventi meteorologici estremi.

La portabilità, i requisiti di potenza inferiori e l'analisi in tempo reale abilitati da TinyML lo rendono adatto ad affrontare diverse sfide di sostenibilità che le regioni in via di sviluppo devono affrontare. L'ampia distribuzione di soluzioni di alimentazione ha il potenziale per fornire un monitoraggio localizzato e conveniente per aiutare a raggiungere alcuni degli Obiettivi di Sviluppo Sostenibile delle Nazioni Unite. Nelle restanti sezioni, approfondiremo il modo in cui TinyML è utile in molti settori che possono affrontare le SDG delle Nazioni Unite.

19.2 Agricoltura

L'agricoltura è essenziale per raggiungere molti degli obiettivi di sviluppo sostenibile delle Nazioni Unite, tra cui l'eradicazione della fame e della malnutrizione, la promozione della crescita economica e l'uso sostenibile delle risorse naturali. TinyML può essere uno strumento prezioso per aiutare a promuovere un'agricoltura sostenibile, in particolare per i piccoli agricoltori nelle regioni in via di sviluppo.

Le soluzioni TinyML possono fornire monitoraggio in tempo reale e analisi dei dati per la salute delle colture e le condizioni di crescita, il tutto senza dipendere dall'infrastruttura di connettività. Ad esempio, i moduli di telecamere a basso costo collegati ai microcontrollori possono monitorare malattie, parassiti e carenze nutrizionali. Gli algoritmi TinyML possono analizzare le immagini per rilevare i problemi in anticipo prima che si diffondano e danneggino i raccolti. Il monitoraggio di precisione può ottimizzare input come acqua, fertilizzanti e pesticidi, migliorando l'efficienza e la sostenibilità.

Altri sensori, come unità GPS e accelerometri, possono tracciare le condizioni del microclima, l'umidità del suolo e il benessere del bestiame. I dati locali in tempo reale aiutano gli agricoltori a rispondere e ad adattarsi meglio ai cambiamenti sul campo. L'analisi TinyML nell'edge evita ritardi, interruzioni di rete e gli elevati costi dei dati dei sistemi basati su cloud. I sistemi localizzati consentono la personalizzazione di colture, malattie e problemi regionali specifici.

Le applicazioni TinyML diffuse possono aiutare a digitalizzare le piccole aziende agricole per aumentare produttività, redditi e resilienza. Il basso costo dell'hardware e i requisiti minimi di connettività rendono le soluzioni accessibili. I progetti nei paesi in via di sviluppo hanno mostrato i vantaggi:

- Il progetto FarmBeats di Microsoft è un approccio end-to-end per abilitare l'agricoltura basata sui dati utilizzando sensori a basso costo, droni e algoritmi di visione e apprendimento automatico. Il progetto cerca di risolvere il problema dell'adozione limitata della tecnologia nell'agricoltura a causa della necessità di maggiore potenza e connettività Internet nelle aziende agricole e della limitata competenza tecnologica degli agricoltori. Il progetto tenta di aumentare la produttività agricola e ridurre i costi

accoppiando i dati con la conoscenza e l'intuizione degli agricoltori sulle loro aziende agricole. Il progetto ha consentito con successo di ottenere informazioni fruibili dai dati tramite la creazione di modelli di intelligenza artificiale (IA) o apprendimento automatico (ML) basati su set di dati fusi. Figura 19.2 illustra il funzionamento interno di FarmBeats di Microsoft.

- Nell'Africa subsahariana, telecamere standard e IA edge hanno ridotto le perdite per malattie della manioca dal 40% al 5%, proteggendo una coltura di base (Ramcharan et al. 2017).
- In Indonesia, i sensori monitorano i microclimi nelle risaie, ottimizzando l'uso dell'acqua anche in caso di piogge irregolari (Tirtalistyani, Murtiningrum, e Kanwar 2022).

Con maggiori investimenti e integrazione nei servizi di consulenza rurale, TinyML potrebbe trasformare l'agricoltura su piccola scala e migliorare i mezzi di sostentamento degli agricoltori in tutto il mondo. La tecnologia porta efficacemente i vantaggi dell'agricoltura di precisione alle regioni disconnesse più bisognose.

Figura 19.2: Microsoft Farmbeats consente di prendere decisioni basate sui dati per migliorare la resa agricola, ridurre i costi complessivi e ridurre l'impatto ambientale della produzione agricola. Fonte: MOLD

Esercizio 19.1: Modellazione della Resa delle Colture

Questo esercizio insegna come prevedere le rese delle colture in Nepal combinando dati satellitari (Sentinel-2), dati climatici (WorldClim) e misure sul campo. Si userà un algoritmo di apprendimento automatico chiamato XGBoost Regressor per creare un modello, dividere i dati per l'addestramento e il test e perfezionare i parametri del modello per ottenere le migliori prestazioni. Questo notebook getta le basi per l'implementazione di TinyML nel settore agricolo. Considerare come si potrebbe adattare questo processo per set di dati più piccoli, meno funzionalità e modelli semplificati per renderlo compatibile con i vincoli di potenza e memoria dei dispositivi TinyML.

Open in Colab

19.3 Assistenza Sanitaria

19.3.1 Espansione dell'Accesso

La copertura sanitaria universale e l'assistenza di qualità restano fuori dalla portata di milioni di persone in tutto il mondo. In molte regioni è necessario un numero maggiore di professionisti sanitari per accedere a diagnosi e trattamenti di base. Inoltre, è necessario migliorare le infrastrutture sanitarie come cliniche, ospedali e servizi di pubblica utilità per alimentare apparecchiature complesse. Queste lacune hanno un impatto sproporzionato sulle comunità emarginate, esacerbando le disparità sanitarie.

TinyML offre una promettente soluzione tecnologica per aiutare ad ampliare l'accesso a un'assistenza sanitaria di qualità a livello globale. TinyML si riferisce alla capacità di implementare algoritmi di apprendimento automatico su microcontrollori, piccoli chip con potenza di elaborazione, memoria e connettività limitate. TinyML consente l'analisi dei dati in tempo reale e l'intelligenza in dispositivi compatti e a bassa potenza.

Ciò crea opportunità per strumenti medici trasformativi che sono portatili, convenienti e accessibili. Il software e l'hardware TinyML possono essere ottimizzati per funzionare anche in ambienti con risorse limitate. Ad esempio, un sistema TinyML potrebbe analizzare i sintomi o fare previsioni diagnostiche utilizzando una potenza di calcolo minima, nessuna connettività Internet continua e una batteria o una fonte di energia solare. Queste capacità possono portare screening e monitoraggio di livello medico direttamente ai pazienti meno assistiti.

19.3.2 Diagnosi Precoce

La diagnosi precoce delle malattie è una delle principali applicazioni. Piccoli sensori abbinati al software TinyML possono identificare i sintomi prima che le condizioni peggiorino o compaiano segni visibili. Ad esempio, i monitoratori della tosse con apprendimento automatico embedded possono rilevare pattern acustici indicativi di malattie respiratorie, malaria o tubercolosi. Rilevare le malattie all'esordio migliora i risultati e riduce i costi sanitari.

Un esempio dettagliato potrebbe essere il monitoraggio della polmonite nei bambini da parte di TinyML. La polmonite è una delle principali cause di morte nei bambini sotto i 5 anni e rilevarla precocemente è fondamentale. Una startup chiamata Respira xColabs ha sviluppato un sensore audio indossabile a basso costo che utilizza algoritmi TinyML per analizzare la tosse e identificare i sintomi di malattie respiratorie come la polmonite. Il dispositivo contiene un microfono e un microcontrollore che esegue un modello di rete neurale addestrato per classificare i suoni respiratori. Può identificare caratteristiche come respiro sibilante, crepitio e stridore che possono indicare la polmonite. Il dispositivo è progettato per essere altamente accessibile: ha un semplice cinturino, non richiede batteria o ricarica e i risultati vengono forniti tramite luci LED e segnali acustici.

Un altro esempio riguarda i ricercatori dell'UNIFEI in Brasile che hanno sviluppato un dispositivo a basso costo che sfrutta TinyML per monitorare i ritmi cardiaci. La loro soluzione innovativa risponde a un'esigenza critica: la fibrillazione atriale e altre anomalie del ritmo cardiaco spesso non vengono diagnosticate a causa del costo proibitivo e della limitata disponibilità di strumenti di screening. Il dispositivo supera queste barriere grazie al suo design ingegnoso. Utilizza un microcontrollore standard che costa solo pochi dollari, insieme a un sensore di pulsazioni di base. Riducendo al minimo la complessità, il dispositivo diventa accessibile alle popolazioni con risorse insufficienti. L'algoritmo TinyML in esecuzione localmente sul microcontrollore analizza i dati delle pulsazioni in tempo reale per rilevare ritmi cardiaci irregolari. Questo dispositivo salvavita per il monitoraggio cardiaco dimostra

come TinyML consenta di implementare potenti capacità di intelligenza artificiale in progetti convenienti e intuitivi.
La versatilità di TinyML promette anche di affrontare le malattie infettive. I ricercatori hanno proposto di applicare TinyML per identificare le zanzare che diffondono la malaria tramite i suoni del battito delle ali. Se dotati di microfoni, i piccoli microcontrollori possono eseguire modelli avanzati di classificazione audio per determinare le specie di zanzare. Questa soluzione compatta e a basso consumo produce risultati in tempo reale, adatti per l'uso in campo remoto. Rendendo l'analisi entomologica conveniente e accessibile, TinyML potrebbe rivoluzionare il monitoraggio degli insetti che mettono a rischio la salute umana. TinyML sta ampliando l'accesso all'assistenza sanitaria per le comunità vulnerabili, dalle malattie cardiache alla malaria.

19.3.3 Controllo delle Malattie Infettive

Le zanzare rimangono il vettore di malattie più mortale al mondo, trasmettendo malattie che infettano oltre un miliardo di persone ogni anno («Vector-borne diseases», s.d.). Malattie come la malaria, la dengue e lo Zika sono particolarmente diffuse nelle regioni con risorse limitate e prive di infrastrutture solide per il controllo delle zanzare. Il monitoraggio delle popolazioni locali di zanzare è essenziale per prevenire le epidemie e indirizzare correttamente gli interventi.
I metodi di monitoraggio tradizionali sono costosi, richiedono molta manodopera e sono difficili da implementare da remoto. La soluzione TinyML proposta supera queste barriere. Piccoli microfoni abbinati ad algoritmi di apprendimento automatico possono classificare le zanzare per specie in base a piccole differenze nelle oscillazioni delle ali. Il software TinyML funziona in modo efficiente su microcontrollori a basso costo, eliminando la necessità di connettività continua.
Un team di ricerca collaborativo dell'Università di Khartoum e dell'ICTP sta esplorando una soluzione innovativa utilizzando TinyML. In un recente articolo, hanno presentato un dispositivo a basso costo in grado di identificare le specie di zanzare che diffondono malattie attraverso i suoni del battito delle ali (Altayeb, Zennaro, e Rovai 2022).
Questo sistema portatile e autonomo promette molto bene per l'entomologia. I ricercatori suggeriscono che potrebbe rivoluzionare le strategie di monitoraggio degli insetti e di controllo dei vettori nelle aree remote. TinyML potrebbe rafforzare significativamente gli sforzi di eradicazione della malaria fornendo analisi delle zanzare più economiche e semplici. La sua versatilità e il fabbisogno energetico minimo lo rendono ideale per l'uso sul campo in regioni isolate e fuori dalla rete con risorse scarse ma un elevato carico di malattie.

19.3.4 TinyML Design Contest in Healthcare

Il primo concorso TinyML in ambito sanitario, TDC'22 (Zhenge Jia et al. 2023), si è tenuto nel 2022 per motivare i team partecipanti a progettare algoritmi IA/ML per rilevare aritmie ventricolari (VA) potenzialmente letali e distribuirli su "Implantable Cardioverter Defibrillators (ICDs)". Le VA sono la causa principale di "sudden cardiac death (SCD)" [morte cardiaca improvvisa]. Le persone ad alto rischio di SCD si affidano all'ICD per erogare un trattamento di defibrillazione adeguato e tempestivo (ad esempio, riportando il cuore al ritmo normale) quando soffrono di VA potenzialmente letali.
Un algoritmo sul dispositivo per il rilevamento precoce e tempestivo di VA potenzialmente letali aumenterà le possibilità di sopravvivenza. L'algoritmo IA/ML proposto doveva essere distribuito ed eseguito su un microcontrollore (MCU) a bassissimo consumo energetico e risorse limitate (una scheda di sviluppo da $10 con un core ARM Cortex-M4 a 80 MHz, 256 kB di memoria flash e 64 kB di SRAM). I progetti presentati sono stati valutati tramite parametri

misurati sull'MCU per (1) prestazioni di rilevamento, (2) latenza di inferenza e (3) occupazione di memoria da parte del programma di algoritmi IA/ML.
Il campione, GaTech EIC Lab, ha ottenuto 0,972 in F_β (punteggio F1 con un peso maggiore da richiamare), 1,747 ms di latenza e 26,39 kB di footprint di memoria con una rete neurale profonda. Un ICD con un algoritmo di rilevamento VA sul dispositivo è stato impiantato in uno studio clinico.

Esercizio 19.2: Dati Clinici: Sbloccare Informazioni con il Riconoscimento di Entità Denominate

In questo esercizio, si imparerè il"Named Entity Recognition (NER)" [riconoscimento di entità denominate], un potente strumento per estrarre informazioni preziose dal testo clinico. Utilizzando Spark NLP, una libreria specializzata per NLP sanitaria, esploreremo come i modelli NER come BiLSTM-CNN-Char e BERT possono identificare automaticamente importanti entità mediche come diagnosi, farmaci, risultati di test e altro ancora. Si acquisirà esperienza pratica applicando queste tecniche con un'attenzione particolare all'estrazione di dati correlati all'oncologia, aiutandoti a sbloccare informazioni sui tipi di cancro e sui dettagli del trattamento dalle cartelle cliniche dei pazienti.

Open in Colab

19.4 Scienza

In molti campi scientifici, i ricercatori sono limitati dalla qualità e dalla risoluzione dei dati che possono raccogliere. Spesso devono dedurre indirettamente i veri parametri di interesse utilizzando correlazioni approssimative e modelli basati su punti dati sparsi. Ciò limita l'accuratezza della comprensione scientifica e delle previsioni.
L'emergere di TinyML apre nuove possibilità per la raccolta di misurazioni scientifiche ad alta fedeltà. Con l'apprendimento automatico embedded, piccoli sensori a basso costo possono elaborare e analizzare automaticamente i dati localmente in tempo reale. Ciò crea reti di sensori intelligenti che catturano dati sfumati su scale e frequenze molto più grandi.
Ad esempio, il monitoraggio delle condizioni ambientali per modellare il cambiamento climatico rimane una sfida a causa della necessità di dati diffusi e continui. Il progetto Ribbit dell'UC Berkeley è pioniere di una soluzione TinyML crowdsourcing (Rao 2021). Hanno sviluppato un sensore CO_2 open source che utilizza un microcontrollore integrato per elaborare le misurazioni del gas. Un set di dati esteso può essere aggregato distribuendo centinaia di questi sensori a basso costo. I dispositivi TinyML compensano i fattori ambientali e forniscono letture granulari, accurate e in precedenza impossibili.
Il potenziale di scalare massicciamente il rilevamento intelligente tramite TinyML ha profonde implicazioni scientifiche. I dati ad alta risoluzione possono portare a scoperte e capacità predittive in campi che vanno dall'ecologia alla cosmologia. Altre applicazioni potrebbero includere sensori sismici per sistemi di allerta precoce sui terremoti, monitor meteorologici distribuiti per tracciare i cambiamenti del microclima e sensori acustici per studiare le popolazioni animali.
Man mano che i sensori e gli algoritmi continuano a migliorare, le reti TinyML potrebbero generare mappe più dettagliate che mai dei sistemi naturali. Democratizzare la raccolta di dati scientifici può accelerare la ricerca e la comprensione tra le discipline. Tuttavia, solleva nuove sfide in merito alla qualità dei dati, alla privacy e alla modellazione di incognite. TinyML indica una crescente convergenza tra intelligenza artificiale e scienze naturali per rispondere a domande fondamentali.

19.5 Conservazione e Ambiente

TinyML sta emergendo come un potente strumento per la conservazione ambientale e gli sforzi di sostenibilità. Una ricerca recente ha evidenziato numerose applicazioni del tiny machine learning in ambiti quali il monitoraggio della fauna selvatica, la gestione delle risorse naturali e il monitoraggio dei cambiamenti climatici.

Un esempio è l'utilizzo di TinyML per il monitoraggio e la protezione della fauna selvatica in tempo reale. I ricercatori hanno sviluppato dispositivi Smart Wildlife Tracker che sfruttano gli algoritmi TinyML per rilevare le attività di bracconaggio. I collari contengono sensori come telecamere, microfoni e GPS per monitorare costantemente l'ambiente circostante. I modelli di machine learning embedded analizzano i dati audio e visivi per identificare minacce come esseri umani nelle vicinanze o spari. Il rilevamento precoce del bracconaggio fornisce alle guardie forestali informazioni fondamentali per intervenire e agire.

Altri progetti applicano TinyML per studiare il comportamento degli animali tramite sensori. Il collare intelligente per la fauna selvatica utilizza accelerometri e monitoraggio acustico per tracciare i movimenti, la comunicazione e gli stati d'animo degli elefanti (T. D. S. Verma 2022). I dispositivi collare TinyML a basso consumo energetico trasmettono dati approfonditi sulle attività degli elefanti evitando gravosi cambi di batteria. Ciò aiuta i ricercatori a osservare in modo discreto le popolazioni di elefanti per informare sulle strategie di conservazione.

Su scala più ampia, i dispositivi TinyML distribuiti sono concepiti per creare reti di sensori dense per la modellazione ambientale. Centinaia di monitor della qualità dell'aria a basso costo potrebbero mappare l'inquinamento nelle città. I sensori sottomarini potrebbero rilevare le tossine e dare un avviso precoce di fioriture algali. Tali applicazioni sottolineano la versatilità di TinyML in ecologia, climatologia e sostenibilità.

I ricercatori della Moulay Ismail University di Meknes in Marocco (Bamoumen et al. 2022) hanno pubblicato un sondaggio su come TinyML può essere utilizzato per risolvere i problemi ambientali. Tuttavia, valutare attentamente i benefici, i rischi e l'accesso equo sarà fondamentale man mano che TinyML espande la ricerca e la conservazione ambientale. Con una considerazione etica degli impatti, TinyML offre soluzioni basate sui dati per proteggere la biodiversità, le risorse naturali e il nostro pianeta.

19.6 Risposta ai Disastri

Nella risposta ai disastri, velocità e sicurezza sono fondamentali. Ma macerie e rottami creano ambienti pericolosi e angusti che ostacolano gli sforzi di ricerca umani. TinyML consente ai droni agili di assistere le squadre di soccorso in questi scenari pericolosi. L'elaborazione locale dei dati tramite TinyML consente una rapida interpretazione per guidare i soccorsi.

Quando gli edifici crollano dopo i terremoti, i piccoli droni possono rivelarsi preziosi. Dotati di algoritmi di navigazione TinyML, i droni di piccole dimensioni come il CrazyFlie con meno di 200 KB di RAM e una frequenza di clock della CPU di soli 168 MHz possono attraversare in sicurezza spazi angusti e mappare percorsi oltre la portata umana (Bardienus P. Duisterhof et al. 2019). L'elusione degli ostacoli consente a questi droni di muoversi tra detriti instabili. Questa mobilità autonoma consente loro di spazzare rapidamente aree in cui gli umani non possono accedere. Fondamentalmente, i sensori di bordo e i processori TinyML analizzano i dati in tempo reale per identificare i segni dei sopravvissuti. Le telecamere termiche possono rilevare il calore corporeo, i microfoni possono captare le richieste di aiuto e i sensori di gas possono avvisare di perdite (Bardienus P. Duisterhof et al. 2021).

Video 19.1 mostra come l'apprendimento di rinforzo profondo può essere utilizzato per consentire ai droni di cercare autonomamente fonti di luce.

Video 19.1: Imparare a Cercare

https://www.youtube.com/watch?v=wmVKbX7MOnU

Video 19.2 è una panoramica dei droni autonomi per il rilevamento delle perdite di gas.

Video 19.2

https://www.youtube.com/watch?v=hj_SBSpK5qg

Inoltre, gli sciami coordinati di droni sbloccano nuove capacità. Collaborando e condividendo informazioni, i team di droni hanno una visione completa della situazione. La copertura dei siti del disastro consente agli algoritmi TinyML di fondere e analizzare i dati da più punti di vista, amplificando la consapevolezza della situazione oltre i singoli droni (Bardienus P. Duisterhof et al. 2021).

Ancora più importante, la ricognizione iniziale dei droni aumenta la sicurezza per i soccorritori umani. Mantenere le squadre di soccorso a una distanza di sicurezza finché i rilievi dei droni non valutano i pericoli salva vite umane. Una volta protetti, i droni possono guidare il posizionamento preciso del personale.

Combinando mobilità agile, dati in tempo reale e coordinamento dello sciame, i droni abilitati TinyML promettono di trasformare la risposta ai disastri. La loro versatilità, velocità e sicurezza li rendono una risorsa vitale per gli sforzi di soccorso in ambienti pericolosi e inaccessibili. L'integrazione di droni autonomi con metodi tradizionali può accelerare le risposte quando è più importante.

19.7 Istruzione e Sensibilizzazione

TinyML ha un immenso potenziale per aiutare ad affrontare le sfide nelle regioni in via di sviluppo, ma per realizzare i suoi benefici è necessaria un'istruzione mirata e un rafforzamento delle capacità. Riconoscendo questa necessità, i ricercatori accademici hanno guidato iniziative di sensibilizzazione per diffondere l'istruzione TinyML a livello globale.

Nel 2020, l'Università di Harvard, la Columbia University, l'International Centre for Theoretical Physics (ICTP) e l'UNIFEI hanno fondato congiuntamente la rete "TinyML for Developing Communities (TinyML4D)" (Zennaro, Plancher, e Reddi 2022). Questa rete consente alle università e ai ricercatori nei paesi in via di sviluppo di sfruttare TinyML per un impatto locale.

Un obiettivo fondamentale è l'espansione dell'accesso all'istruzione applicata all'apprendimento automatico. La rete TinyML4D fornisce formazione, programmi di studio e risorse di laboratorio ai membri. Workshop pratici e progetti di raccolta dati offrono agli studenti esperienza pratica. I membri possono condividere le best practice e creare una comunità attraverso conferenze e collaborazioni accademiche.

La rete dà priorità all'abilitazione di soluzioni TinyML localmente rilevanti. I progetti affrontano sfide come agricoltura, salute e monitoraggio ambientale in base alle esigenze della comunità. Ad esempio, un'università membro in Ruanda ha sviluppato un sistema di monitoraggio delle inondazioni a basso costo utilizzando TinyML e sensori.

TinyML4D include oltre 50 istituzioni membri in Africa, Asia e America Latina. Tuttavia, sono necessari maggiori investimenti e partnership industriali per raggiungere tutte le regioni sottoservite. La visione finale è quella di formare le nuove generazioni ad applicare eticamente TinyML per uno sviluppo sostenibile. Gli sforzi di sensibilizzazione odierni gettano le basi per democratizzare la tecnologia trasformativa per il futuro.

19.8 Accessibilità

La tecnologia ha un potenziale immenso per abbattere le barriere affrontate dalle persone con disabilità e colmare le lacune nell'accessibilità. TinyML apre specificamente nuove possibilità per lo sviluppo di dispositivi di assistenza intelligenti e personalizzati.

Con algoritmi di apprendimento automatico eseguiti localmente su microcontrollori, gli strumenti di accessibilità compatti possono funzionare in tempo reale senza dipendere dalla connettività. Il National Institute on Deafness and Other Communication Disorders (NIDCD) afferma che il 20% della popolazione mondiale ha una qualche forma di perdita dell'udito. Gli apparecchi acustici che sfruttano TinyML potrebbero riconoscere più parlanti e amplificare la voce di un target scelto in stanze affollate. Ciò consente alle persone con problemi di udito di concentrarsi su conversazioni specifiche.

Analogamente, i dispositivi di mobilità potrebbero utilizzare l'elaborazione della vista sul dispositivo per identificare ostacoli e caratteristiche del terreno. Ciò consente una navigazione e una sicurezza migliorate per gli ipovedenti. Aziende come Envision stanno sviluppando occhiali intelligenti, convertendo le informazioni visive in parlato, con TinyML embedded per guidare le persone non vedenti rilevando oggetti, testo e segnali stradali. Video 19.3 di seguito mostra i diversi casi di utilizzo nella vita reale degli occhiali per l'ausilio visivo Envision.

Video 19.3

https://www.youtube.com/watch?v=oGWinIKDOdc

TinyML potrebbe persino alimentare arti protesici reattivi. Analizzando i segnali nervosi e i dati sensoriali come la tensione muscolare, protesi ed esoscheletri con ML embedded possono muoversi e regolare la presa in modo dinamico, rendendo il controllo più naturale e intuitivo. Le aziende stanno creando mani bioniche economiche e per uso quotidiano utilizzando TinyML. Per coloro che hanno difficoltà di linguaggio, i dispositivi abilitati alla voce con TinyML possono generare output vocali personalizzati da input non verbali. Pairs di Anthropic traduce i gesti in un linguaggio naturale su misura per i singoli utenti.

Abilitando una tecnologia assistiva più personalizzabile, TinyML rende i servizi più accessibili e su misura per le esigenze individuali. E attraverso applicazioni di traduzione e interpretazione, TinyML può abbattere le barriere comunicative. App come Microsoft Translator offrono traduzioni in tempo reale basate sugli algoritmi TinyML.

Con il suo design ponderato e inclusivo, TinyML promette più autonomia e dignità per le persone con disabilità. Tuttavia, gli sviluppatori dovrebbero coinvolgere direttamente le comunità, evitare di compromettere la privacy e considerare l'accessibilità economica per massimizzare i benefici. TinyML ha un enorme potenziale per contribuire a un mondo più giusto ed equo.

19.9 Infrastruttura e Pianificazione Urbana

Con l'aumento della popolazione urbana, le città affrontano sfide immense nella gestione efficiente di risorse e infrastrutture. TinyML presenta un potente strumento per lo sviluppo di sistemi intelligenti per ottimizzare le operazioni e la sostenibilità della città. Potrebbe rivoluzionare l'efficienza energetica negli edifici intelligenti.

I modelli di apprendimento automatico possono imparare a prevedere e regolare l'uso di energia in base ai pattern di occupazione. I sensori miniaturizzati posizionati negli edifici possono fornire dati granulari e in tempo reale sull'utilizzo dello spazio, sulla temperatura e altro ancora (Seyedzadeh et al. 2018). Questa visibilità consente ai sistemi TinyML di ridurre al minimo gli sprechi ottimizzando riscaldamento, raffreddamento, illuminazione, ecc.

Questi esempi dimostrano l'enorme potenziale di TinyML per infrastrutture cittadine efficienti e sostenibili. Tuttavia, gli urbanisti devono considerare privacy, sicurezza e accessibilità per garantire un'adozione responsabile. Con un'implementazione attenta, TinyML potrebbe modernizzare profondamente la vita urbana.

19.10 Sfide e Considerazioni

Sebbene TinyML offra immense opportunità, sarà fondamentale considerare attentamente le sfide e le implicazioni etiche man mano che l'adozione si diffonde a livello globale. I ricercatori hanno evidenziato i fattori chiave da affrontare, soprattutto quando si distribuisce TinyML nelle regioni in via di sviluppo.

Una delle sfide più importanti è l'accesso limitato alla formazione e all'hardware (Ooko et al. 2021). Esistono solo programmi educativi su misura per TinyML e le economie emergenti hanno spesso bisogno di una solida catena di fornitura di elettronica. Saranno necessarie una formazione approfondita e partnership per coltivare le competenze e rendere i dispositivi disponibili alle comunità svantaggiate. Iniziative come la rete TinyML4D aiutano a fornire percorsi di apprendimento strutturati.

Anche le limitazioni dei dati pongono ostacoli. I modelli TinyML richiedono set di dati localizzati di qualità, che sono scarsi in ambienti con risorse insufficienti. La creazione di framework per il crowdsourcing dei dati in modo etico potrebbe risolvere questo problema. Tuttavia, la raccolta dei dati dovrebbe avvantaggiare direttamente le comunità locali, non solo estrarre valore.

L'ottimizzazione dell'uso dell'energia e della connettività sarà fondamentale per la sostenibilità. Le basse esigenze di potenza di TinyML lo rendono ideale per casi d'uso fuori dalla rete. L'integrazione di batterie o energia solare può consentire un funzionamento continuo. Adattare i dispositivi per la trasmissione a bassa larghezza di banda in cui Internet è limitato massimizza anche l'impatto.

Le barriere culturali e linguistiche complicano ulteriormente l'adozione. Le interfacce utente e i dispositivi dovrebbero tenere conto di tutti i livelli di alfabetizzazione ed evitare di escludere sottogruppi. Le soluzioni controllabili vocalmente nei dialetti locali possono migliorare l'accessibilità.

Affrontare queste sfide richiede partnership olistiche, finanziamenti e supporto politico. Tuttavia, l'inclusione e l'etica di TinyML hanno un potenziale monumentale per elevare le popolazioni svantaggiate in tutto il mondo. Con un'implementazione ponderata, la tecnologia potrebbe democratizzare profondamente le opportunità.

19.11 Conclusione

TinyML offre un'enorme opportunità di sfruttare la potenza dell'intelligenza artificiale per promuovere gli Obiettivi di sviluppo sostenibile delle Nazioni Unite e guidare l'impatto sociale a livello globale, come evidenziato da esempi in settori quali sanità, agricoltura, conservazione e altro; l'apprendimento automatico embedded sblocca nuove capacità per soluzioni accessibili e a basso costo, su misura per i contesti locali. TinyML aggira barriere come infrastrutture scadenti, connettività limitata e costi elevati che spesso escludono le comunità in via di sviluppo dalle tecnologie emergenti.

Tuttavia, realizzare il pieno potenziale di TinyML richiede una collaborazione olistica. Ricercatori, politici, aziende e stakeholder locali devono collaborare per fornire formazione, stabilire quadri etici, progettare soluzioni in comune e adattarle alle esigenze della comunità. Attraverso uno sviluppo e un'implementazione inclusivi, TinyML può mantenere la promessa di colmare le disuguaglianze e migliorare le popolazioni vulnerabili senza lasciare indietro nessuno.

Se coltivato in modo responsabile, TinyML potrebbe democratizzare le opportunità e accelerare i progressi sulle priorità globali, dall'alleviamento della povertà alla resilienza climatica. La tecnologia rappresenta una nuova ondata di IA applicata per potenziare le società, promuovere la sostenibilità e spingere l'umanità verso una maggiore giustizia, prosperità e pace. TinyML offre uno sguardo a un futuro abilitato dall'IA accessibile a tutti.

19.12 Risorse

Ecco un elenco curato di risorse per supportare studenti e insegnanti nei loro percorsi di apprendimento e insegnamento. Stiamo lavorando costantemente per espandere questa raccolta e presto aggiungeremo nuovi esercizi.

Slide

Queste slide sono uno strumento prezioso per gli insegnanti per tenere lezioni e per gli studenti per rivedere il materiale secondo il proprio ritmo. Incoraggiamo studenti e docenti a sfruttare queste slide per migliorare la loro comprensione e facilitare un trasferimento efficace delle conoscenze.

- TinyML for Social Impact.

Video

- Video 19.1
- Video 19.2
- Video 19.3

Esercizi

- Esercizio 19.1
- Esercizio 19.2

Laboratori

Oltre agli esercizi, offriamo una serie di laboratori pratici che consentono agli studenti di acquisire esperienza pratica con le tecnologie di intelligenza artificiale embedded. Questi laboratori forniscono una guida passo dopo passo, consentendo agli studenti di sviluppare le proprie competenze in un ambiente strutturato e di supporto. Siamo lieti di annunciare che presto saranno disponibili nuovi laboratori, che arricchiranno ulteriormente l'esperienza di apprendimento.

- *Prossimamente.*

20. Conclusione

DALL·E 3 Prompt: Un'immagine che raffigura l'ultimo capitolo di un libro sui sistemi ML, aperto su due pagine. Le pagine riassumono concetti chiave come reti neurali, architetture modello, accelerazione hardware e MLOps. Una pagina presenta un diagramma di una rete neurale e diverse architetture modello, mentre l'altra pagina mostra illustrazioni di componenti hardware per flussi di lavoro di accelerazione e MLOps. Lo sfondo include elementi vaghi come schemi di circuiti e dati per rafforzare il tema tecnologico. I colori sono professionali e puliti, con un'enfasi sulla chiarezza e sulla comprensione.

20.1 Introduzione

Questo libro esamina il campo in rapida evoluzione dei sistemi ML (Capitolo 2). Ci siamo concentrati sui sistemi perché, nonostante esistano numerose risorse sui modelli e sugli algoritmi di ML, c'è ancora molto da capire su come costruire i sistemi che li eseguono.

Per fare un'analogia, consideriamo il processo di costruzione di un'auto. Sebbene siano disponibili molte risorse sui vari componenti di un'auto, come motore, trasmissione e sospensioni, spesso è necessario comprendere meglio come assemblare questi componenti in un veicolo funzionale. Proprio come un'auto richiede un sistema ben progettato e correttamente integrato per funzionare in modo efficiente e affidabile, anche i modelli ML richiedono un sistema robusto e costruito con cura per offrire il loro pieno potenziale. Inoltre, vi sono molte sfumature nella costruzione di sistemi ML, dato il loro caso d'uso specifico. Ad esempio, un'auto da corsa di Formula 1 deve essere assemblata in modo diverso da una normale auto Prius di tutti i giorni.

Il nostro viaggio è iniziato tracciando la traiettoria storica del ML, dalle sue fondamenta teoriche al suo stato attuale come forza trasformativa in tutti i settori (Capitolo 3). Questo viaggio ha evidenziato i notevoli progressi nel campo, le sfide e le opportunità.

In questo libro, abbiamo esaminato le complessità dei sistemi ML, esaminando i componenti critici e le best practice necessarie per creare una pipeline fluida ed efficiente. Dalla preelaborazione dei dati e dalla formazione del modello alla distribuzione e al monitoraggio, abbiamo fornito approfondimenti e indicazioni per aiutare i lettori a orientarsi nel complesso panorama dello sviluppo del sistema ML.

I sistemi ML coinvolgono flussi di lavoro complessi, che abbracciano vari argomenti, dall'ingegneria dei dati alla distribuzione del modello su sistemi diversi (Capitolo 4). Fornendo una panoramica di questi componenti del sistema ML, abbiamo mirato a mostrare l'enorme profondità e ampiezza del campo e le competenze necessarie. Comprendere le

complessità dei flussi di lavoro di apprendimento automatico è fondamentale sia per i professionisti sia per i ricercatori, poiché consente loro di orientarsi in modo efficace nel panorama e di sviluppare soluzioni di apprendimento automatico solide, efficienti e di impatto.

Concentrandoci sull'aspetto sistemico del ML, puntiamo a colmare il divario tra conoscenza teorica e implementazione pratica. Proprio come un sistema corporeo umano sano consente agli organi di funzionare in modo ottimale, un sistema ML ben progettato consente ai modelli di fornire costantemente risultati accurati e affidabili. L'obiettivo di questo libro è quello di fornire ai lettori le conoscenze e gli strumenti necessari per creare sistemi ML che mostrino la potenza dei modelli sottostanti e garantiscano un'integrazione e un funzionamento fluidi, proprio come un corpo umano ben funzionante.

20.2 Conoscere l'Importanza dei Dataset ML

Uno degli aspetti chiave che abbiamo sottolineato è che i dati sono la base su cui sono costruiti i sistemi ML (Capitolo 5). I dati sono il nuovo codice che programma reti neurali profonde, rendendo l'ingegneria dei dati la prima e più critica fase di qualsiasi pipeline ML. Ecco perché abbiamo iniziato la nostra esplorazione immergendoci nelle basi dell'ingegneria dei dati, riconoscendo che qualità, diversità e approvvigionamento etico sono fondamentali per creare modelli di apprendimento automatico solidi e affidabili.

L'importanza di dati di alta qualità deve essere bilanciata. Le carenze nella qualità dei dati possono portare a conseguenze negative significative, come previsioni errate, cessazioni di progetti e persino potenziali danni alle comunità. Questi effetti a cascata, spesso chiamati "Data Cascades" [cascate di dati], evidenziano la necessità di pratiche diligenti di gestione e governance dei dati. I professionisti del ML devono dare priorità alla qualità dei dati, garantire diversità e rappresentatività e aderire a standard etici di raccolta e utilizzo dei dati. In questo modo, possiamo mitigare i rischi associati alla scarsa qualità dei dati e creare sistemi ML affidabili, sicuri e vantaggiosi per la società.

20.3 Esplorare il Panorama dei Framework di IA

Esistono molti framework ML diversi. Pertanto, ci siamo immersi nell'evoluzione di diversi framework ML, analizzando il funzionamento interno di quelli più popolari come TensorFlow e PyTorch e fornendo approfondimenti sui componenti principali e sulle funzionalità avanzate che li definiscono (Capitolo 6). Abbiamo anche esaminato la specializzazione di framework su misura per esigenze specifiche, come quelli progettati per l'IA embedded. Abbiamo discusso i criteri per la selezione del framework più adatto per un determinato progetto.

La nostra esplorazione ha anche toccato le tendenze future che dovrebbero plasmare il panorama dei framework ML nei prossimi anni. Man mano che il campo continua a evolversi, possiamo prevedere l'emergere di framework più specializzati e ottimizzati che soddisfano i requisiti unici di diversi domini e scenari di distribuzione, come abbiamo visto con TensorFlow Lite per microcontrollori. Restando al passo con questi sviluppi e comprendendo i compromessi coinvolti nella selezione del framework, possiamo prendere decisioni informate e sfruttare gli strumenti più appropriati per creare sistemi ML efficienti.

Inoltre, ci aspettiamo di vedere una crescente enfasi sull'interoperabilità dei framework e sugli sforzi di standardizzazione, come il formato ONNX (Open Neural Network Exchange). Questo formato consente di addestrare i modelli in un framework e distribuirli in un altro, facilitando una maggiore collaborazione e portabilità su diverse piattaforme e ambienti.

20.4 Comprendere i Fondamenti del Training ML

Come professionisti ML che creano sistemi ML, è fondamentale comprendere a fondo il processo di addestramento dell'IA e le sfide del sistema nel ridimensionarlo e ottimizzarlo.

Sfruttando le capacità dei moderni framework di IA e restando aggiornati con gli ultimi progressi nelle tecniche di training, possiamo creare sistemi ML robusti, efficienti e scalabili in grado di affrontare problemi del mondo reale e guidare l'innovazione in vari domini.
Abbiamo iniziato esaminando i fondamenti della formazione AI (Capitolo 7), che comporta l'inserimento di dati nei modelli ML e la regolazione dei loro parametri per ridurre al minimo la differenza tra output previsti ed effettivi. Questo processo è computazionalmente intensivo e richiede un'attenta considerazione di vari fattori, come la scelta di algoritmi di ottimizzazione, velocità di apprendimento, dimensioni del batch e tecniche di regolarizzazione. Comprendere questi concetti è fondamentale per sviluppare pipeline di training efficaci ed efficienti.
Tuttavia, il training di modelli ML su larga scala pone sfide di sistema significative. Man mano che le dimensioni dei set di dati e la complessità dei modelli aumentano, le risorse computazionali richieste per la formazione possono diventare proibitive. Ciò ha portato allo sviluppo di tecniche di training distribuite, come il parallelismo di dati e di modelli, che consentono a più dispositivi di collaborare nel processo di training. Framework come TensorFlow e PyTorch si sono evoluti per supportare questi paradigmi di training distribuiti, consentendo ai professionisti di scalare i carichi di lavoro di training su cluster di GPU o TPU.
Oltre al training distribuito, abbiamo discusso tecniche per ottimizzare il processo di training, come il training a precisione mista e la compressione del gradiente. È importante notare che, sebbene queste tecniche possano sembrare algoritmiche, hanno un impatto significativo sulle prestazioni del sistema. La scelta degli algoritmi di training, della precisione e delle strategie di comunicazione influisce direttamente sull'utilizzo delle risorse, sulla scalabilità e sull'efficienza del sistema ML. Pertanto, è fondamentale adottare un approccio di co-progettazione algoritmo-hardware o algoritmo-sistema, in cui le scelte algoritmiche vengono effettuate in tandem con le considerazioni del sistema. Comprendendo l'interazione tra algoritmi e hardware, possiamo prendere decisioni informate che ottimizzano le prestazioni del modello e l'efficienza del sistema, portando infine a soluzioni ML più efficaci e scalabili.

20.5 Perseguire l'Efficienza nei Sistemi di IA

L'implementazione di modelli ML addestrati è più complessa della semplice esecuzione delle reti; l'efficienza è fondamentale (Capitolo 8). In questo capitolo sull'efficienza dell'IA, abbiamo sottolineato che l'efficienza non è solo un lusso, ma una necessità nei sistemi di intelligenza artificiale. Abbiamo approfondito i concetti chiave alla base dell'efficienza dei sistemi di IA, riconoscendo che le richieste computazionali sulle reti neurali possono essere scoraggianti, anche per i sistemi minimi. Per integrare perfettamente l'IA nei dispositivi quotidiani e nei sistemi essenziali, deve funzionare in modo ottimale entro i vincoli delle risorse limitate, mantenendo al contempo la sua efficacia.
In tutto il libro, abbiamo evidenziato l'importanza di perseguire l'efficienza per garantire che i modelli di IA siano semplificati, rapidi e sostenibili. Ottimizzando i modelli per l'efficienza, possiamo ampliare la loro applicabilità su varie piattaforme e scenari, consentendo all'IA di essere distribuita in ambienti con risorse limitate come sistemi embedded e dispositivi edge. Questa ricerca dell'efficienza è fondamentale per l'adozione diffusa e l'implementazione pratica delle tecnologie di IA nelle applicazioni del mondo reale.

20.6 Ottimizzazione delle Architetture dei Modelli ML

Abbiamo quindi esplorato varie architetture di modelli, dal perceptron fondamentale alle sofisticate reti di trasformatori, ciascuna adattata a specifiche attività e tipi di dati. Questa esplorazione ha messo in luce la notevole diversità e adattabilità dei modelli di apprendimento automatico, consentendo loro di affrontare vari problemi in tutti i domini.

Tuttavia, quando si distribuiscono questi modelli su sistemi, in particolare sistemi embedded con risorse limitate, l'ottimizzazione del modello diventa una necessità. L'evoluzione delle architetture di modelli, dai primi MobileNet progettati per dispositivi mobili ai più recenti modelli TinyML ottimizzati per microcontrollori, è una testimonianza della continua innovazione.

Nel capitolo sull'ottimizzazione del modello (Capitolo 9), abbiamo esaminato l'arte e la scienza dell'ottimizzazione dei modelli di apprendimento automatico per garantire che siano leggeri, efficienti ed efficaci quando distribuiti in scenari TinyML. Abbiamo esplorato tecniche come la compressione del modello, la quantizzazione e la ricerca dell'architettura, che ci consentono di ridurre l'impronta computazionale dei modelli mantenendone le prestazioni. Applicando queste tecniche di ottimizzazione, possiamo creare modelli su misura per i vincoli specifici dei sistemi embedded, consentendo l'implementazione di potenti capacità di intelligenza artificiale su dispositivi edge. Ciò apre molte possibilità per l'elaborazione e il processo decisionale intelligenti e in tempo reale in applicazioni IoT, robotica e mobile computing. Mentre continuiamo a spingere i confini dell'efficienza dell'intelligenza artificiale, ci aspettiamo di vedere soluzioni ancora più innovative per l'implementazione di modelli di apprendimento automatico in ambienti con risorse limitate.

20.7 Avanzamento dell'Hardware di Elaborazione dell'IA

Nel corso degli anni, abbiamo assistito a notevoli progressi nell'hardware ML, spinti dall'insaziabile domanda di potenza di calcolo e dalla necessità di affrontare le sfide dei vincoli di risorse nelle distribuzioni nel mondo reale (Capitolo 10). Questi progressi sono stati cruciali nel consentire l'implementazione di potenti funzionalità di intelligenza artificiale su dispositivi con risorse limitate, aprendo nuove possibilità in vari settori.

L'accelerazione hardware specializzata è essenziale per superare questi vincoli e abilitare l'apprendimento automatico ad alte prestazioni. Gli acceleratori hardware, come GPU, FPGA e ASIC, ottimizzano le operazioni ad alta intensità di calcolo, in particolare l'inferenza, sfruttando i chip personalizzati progettati per efficienti moltiplicazioni di matrici. Questi acceleratori forniscono sostanziali accelerazioni rispetto alle CPU per uso generico, consentendo l'esecuzione in tempo reale di modelli ML avanzati su dispositivi con rigide limitazioni di dimensioni, peso e potenza.

Abbiamo anche esplorato le varie tecniche e approcci per l'accelerazione hardware nei sistemi di apprendimento automatico embedded. Abbiamo discusso i compromessi nella selezione dell'hardware appropriato per casi d'uso specifici e l'importanza delle ottimizzazioni software per sfruttare appieno le capacità di questi acceleratori. Comprendendo questi concetti, i professionisti del ML possono prendere decisioni informate quando progettano e distribuiscono sistemi ML.

Data la pletora di soluzioni hardware ML disponibili, il benchmarking è diventato essenziale per lo sviluppo e la distribuzione di sistemi di apprendimento automatico (Capitolo 11). Il benchmarking consente agli sviluppatori di misurare e confrontare le prestazioni di diverse piattaforme hardware, architetture di modello, procedure di training e strategie di distribuzione. Utilizzando benchmark consolidati come MLPerf, i professionisti ottengono preziose informazioni sugli approcci più efficaci per un dato problema, considerando i vincoli unici dell'ambiente di distribuzione del target.

I progressi nell'hardware ML, combinati con le informazioni ottenute dalle tecniche di benchmarking e ottimizzazione, hanno aperto la strada alla distribuzione con successo delle capacità di apprendimento automatico su vari dispositivi, dai potenti server edge ai microcontrollori con risorse limitate. Man mano che il campo continua a evolversi, ci aspettiamo di vedere soluzioni hardware e approcci di benchmarking ancora più innovativi

che amplieranno ulteriormente i confini di ciò che è possibile con i sistemi di apprendimento automatico embedded.

20.8 Abbracciare l'Apprendimento "On-Device"

Oltre ai progressi nell'hardware ML, abbiamo anche esplorato l'apprendimento "on-device, in cui i modelli possono adattarsi e apprendere direttamente sul dispositivo (Capitolo 12). Questo approccio ha implicazioni significative per la privacy e la sicurezza dei dati, poiché le informazioni sensibili possono essere elaborate localmente senza la necessità di trasmissione a server esterni.

L'apprendimento "on-device" migliora la privacy mantenendo i dati entro i confini del dispositivo, riducendo il rischio di accessi non autorizzati o violazioni dei dati. Riduce inoltre la dipendenza dalla connettività cloud, consentendo ai modelli ML di funzionare efficacemente anche in scenari con accesso a Internet limitato o intermittente. Abbiamo discusso tecniche come l'apprendimento tramite trasferimento e l'apprendimento federato, che hanno ampliato le capacità dell'apprendimento sul dispositivo. L'apprendimento tramite trasferimento consente ai modelli di sfruttare le conoscenze acquisite da un'attività o dominio per migliorare le prestazioni su un altro, consentendo un apprendimento più efficiente ed efficace su dispositivi con risorse limitate. D'altra parte, l'apprendimento federato consente aggiornamenti collaborativi del modello su dispositivi distribuiti senza aggregazione centralizzata dei dati. Questo approccio consente a più dispositivi di contribuire all'apprendimento mantenendo i propri dati localmente, migliorando la privacy e la sicurezza.

Questi progressi nell'apprendimento su dispositivo hanno aperto la strada ad applicazioni di apprendimento automatico più sicure, rispettose della privacy e decentralizzate. Mentre diamo priorità alla privacy e alla sicurezza dei dati nello sviluppo di sistemi ML, ci aspettiamo di vedere soluzioni più innovative che consentano potenti capacità di IA proteggendo al contempo le informazioni sensibili e garantendo la privacy degli utenti.

20.9 Semplificazione delle Operazioni ML

Anche se abbiamo capito bene i pezzi di cui sopra, sfide e considerazioni devono essere affrontate per garantire un'integrazione e un funzionamento di successo dei modelli ML negli ambienti di produzione. Nel capitolo MLOps (Capitolo 13) abbiamo studiato le pratiche e le architetture necessarie per sviluppare, distribuire e gestire i modelli ML durante il loro intero ciclo di vita. Abbiamo esaminato le fasi di ML, dalla raccolta dati e dal training del modello alla valutazione, distribuzione e monitoraggio continuo.

Abbiamo appreso l'importanza dell'automazione, della collaborazione e del miglioramento continuo in MLOps. Automatizzando i processi chiave, i team possono semplificare i loro flussi di lavoro, ridurre gli errori manuali e accelerare la distribuzione dei modelli ML. La collaborazione tra team diversi, tra cui data scientist, ingegneri ed esperti di dominio, garantisce lo sviluppo e la distribuzione di successo dei sistemi ML.

L'obiettivo finale di questo capitolo era quello di fornire ai lettori una comprensione completa della gestione dei modelli ML, dotandoli delle conoscenze e degli strumenti necessari per creare ed eseguire applicazioni ML che forniscano un valore sostenibile con successo. Adottando le "best practices" in ambito MLOps, le organizzazioni possono garantire il successo e l'impatto a lungo termine delle proprie iniziative di ML, promuovendo l'innovazione e producendo risultati significativi.

20.10 Garantire Sicurezza e Privacy

Nessun sistema ML è mai completo senza pensare a sicurezza e privacy. Sono di fondamentale importanza quando si sviluppano sistemi ML nel mondo reale. Poiché l'apprendimento automatico trova sempre più applicazione in domini sensibili come sanità,

finanza e dati personali, salvaguardare la riservatezza e prevenire l'uso improprio di dati e modelli diventa un imperativo critico, e questi erano i concetti che abbiamo discusso in precedenza (Capitolo 14).

Per creare sistemi ML robusti e responsabili, i professionisti devono comprendere a fondo i potenziali rischi per la sicurezza e la privacy. Questi rischi includono perdite di dati, che possono esporre informazioni sensibili; furto di modelli, in cui attori malintenzionati rubano modelli addestrati; attacchi avversari in grado di manipolare il comportamento del modello; pregiudizi nei modelli che possono portare a risultati ingiusti o discriminatori; e accesso involontario a informazioni private.

Per mitigare questi rischi è necessaria una profonda comprensione delle best practice in materia di sicurezza e privacy. Pertanto, abbiamo sottolineato che la sicurezza e la privacy non possono essere un ripensamento: devono essere affrontate in modo proattivo in ogni fase del ciclo di vita dello sviluppo del sistema ML. Sin dalle fasi iniziali di raccolta ed etichettatura dei dati, è fondamentale garantire che i dati siano gestiti in modo sicuro e che la privacy sia protetta. Durante il training e la valutazione del modello, è possibile impiegare tecniche come la privacy differenziale e il calcolo multi-parte sicuro per salvaguardare le informazioni sensibili.

Quando si distribuiscono modelli ML, è necessario implementare controlli di accesso, crittografia e meccanismi di monitoraggio robusti per impedire l'accesso non autorizzato e rilevare potenziali violazioni della sicurezza. Il monitoraggio e l'audit continui dei sistemi ML come parte di MLOps sono inoltre essenziali per identificare e affrontare le vulnerabilità emergenti di sicurezza o privacy.

Integrando considerazioni sulla sicurezza e sulla privacy in ogni fase di creazione, distribuzione e gestione dei sistemi ML, possiamo sbloccare in modo sicuro i vantaggi dell'IA proteggendo al contempo i diritti degli individui e garantendo l'uso responsabile di queste potenti tecnologie. Solo attraverso questo approccio proattivo e completo possiamo creare sistemi ML che non siano solo tecnologicamente avanzati, ma anche eticamente solidi e degni della fiducia del pubblico.

20.11 Sostenere Considerazioni Etiche

Mentre accogliamo i progressi dell'apprendimento automatico in tutti gli aspetti della nostra vita, è fondamentale tenere a mente le considerazioni etiche che plasmeranno il futuro dell'IA (Capitolo 15). Equità, trasparenza, responsabilità e privacy nei sistemi di IA saranno fondamentali man mano che diventeranno più integrati nelle nostre vite e nei nostri processi decisionali.

Poiché i sistemi di IA stanno diventando sempre più diffusi e influenti, è importante garantire che siano progettati e implementati nel rispetto dei principi etici. Ciò significa mitigare attivamente i pregiudizi, promuovere l'equità e prevenire risultati discriminatori. Inoltre, la progettazione etica dell'IA garantisce la trasparenza nel modo in cui i sistemi di IA prendono decisioni, consentendo agli utenti di comprendere e fidarsi dei loro risultati.

La responsabilità è un'altra considerazione etica fondamentale. Man mano che i sistemi di IA assumono maggiori responsabilità e prendono decisioni che hanno un impatto sugli individui e sulla società, devono esserci meccanismi chiari per ritenere responsabili questi sistemi e i loro creatori. Ciò include la definizione di "framework" per l'audit e il monitoraggio dei sistemi di IA e la definizione di meccanismi di responsabilità e risarcimento in caso di danni o conseguenze indesiderate.

Quadri etici, regolamenti e standard saranno essenziali per affrontare queste sfide etiche. Questi quadri dovrebbero guidare lo sviluppo e l'implementazione responsabili delle tecnologie di IA, assicurando che siano in linea con i valori della società e promuovano il benessere di individui e comunità.

Inoltre, discussioni e collaborazioni in corso tra ricercatori, professionisti, politici e società saranno cruciali per orientarsi nel panorama etico dell'IA. Queste conversazioni dovrebbero essere inclusive e diversificate, riunendo diverse prospettive e competenze per sviluppare soluzioni complete ed eque. Mentre andiamo avanti, è responsabilità collettiva di tutte le parti interessate dare priorità alle considerazioni etiche nello sviluppo e nell'implementazione dei sistemi di IA.

20.12 Promuovere la Sostenibilità e l'Equità

Le crescenti richieste computazionali dell'apprendimento automatico, in particolare per l'addestramento di modelli di grandi dimensioni, hanno sollevato preoccupazioni circa il loro impatto ambientale dovuto all'elevato consumo energetico e alle emissioni di carbonio (Capitolo 16). Man mano che la scala e la complessità dei modelli continuano a crescere, affrontare le sfide di sostenibilità associate allo sviluppo dell'IA diventa imperativo. Per mitigare l'impatto ambientale dell'IA, lo sviluppo di algoritmi efficienti dal punto di vista energetico è fondamentale. Ciò comporta l'ottimizzazione di modelli e procedure di addestramento per ridurre al minimo i requisiti computazionali mantenendo le prestazioni. Tecniche come la compressione del modello, la quantizzazione e la ricerca efficiente dell'architettura neurale possono aiutare a ridurre il consumo energetico dei sistemi di IA. L'utilizzo di fonti di energia rinnovabili per alimentare l'infrastruttura di IA è un altro passo importante verso la sostenibilità. Passando a fonti di energia pulita come quella solare, eolica e idroelettrica, le emissioni di carbonio associate allo sviluppo dell'IA possono essere notevolmente ridotte. Ciò richiede uno sforzo concertato da parte della comunità dell'IA e il supporto di politici e leader del settore per investire e adottare soluzioni di energia rinnovabile. Inoltre, l'esplorazione di paradigmi di elaborazione alternativi, come l'elaborazione neuromorfica e fotonica, promette di sviluppare sistemi di intelligenza artificiale più efficienti dal punto di vista energetico. Sviluppando hardware e algoritmi che emulano i meccanismi di elaborazione del cervello, possiamo potenzialmente creare sistemi di intelligenza artificiale che siano sia potenti che sostenibili.

La comunità dell'intelligenza artificiale deve dare priorità alla sostenibilità come considerazione chiave nella ricerca e nello sviluppo. Ciò implica investire in iniziative di elaborazione ecologica, come lo sviluppo di hardware efficiente dal punto di vista energetico e l'ottimizzazione dei data center per ridurre il consumo di energia. Richiede inoltre la collaborazione tra discipline, riunendo esperti di IA, energia e sostenibilità per sviluppare soluzioni olistiche.

Inoltre, è importante riconoscere che l'accesso alle risorse di elaborazione dell'IA e dell'apprendimento automatico potrebbe non essere distribuito equamente tra organizzazioni e regioni. Questa disparità può portare a un divario crescente tra coloro che hanno i mezzi per sfruttare le tecnologie di IA avanzate e coloro che non li hanno. Organizzazioni come l'Organizzazione per la cooperazione e lo sviluppo economico (OCSE) stanno esplorando attivamente modi per affrontare questo problema e promuovere una maggiore equità nell'accesso e nell'adozione dell'IA. Promuovendo la cooperazione internazionale, condividendo le best practice e supportando iniziative di capacity building, possiamo garantire che i benefici dell'IA siano più ampiamente accessibili e che nessuno venga lasciato indietro nella rivoluzione dell'IA.

20.13 Migliorare la Robustezza e la Resilienza

Il capitolo su IA Robusta approfondisce i concetti fondamentali, le tecniche e gli strumenti per la creazione di sistemi ML fault-tolerant e error-resilient (Capitolo 17). In quel capitolo, abbiamo esplorato come le tecniche di IA robuste possano affrontare le sfide poste da vari

tipi di guasti hardware, inclusi guasti transitori, permanenti e intermittenti, nonché problemi software come bug, difetti di progettazione ed errori di implementazione.
Utilizzando tecniche di IA robuste, i sistemi ML possono mantenere la loro affidabilità, sicurezza e prestazioni anche in condizioni avverse. Queste tecniche consentono ai sistemi di rilevare e ripristinare i guasti, adattarsi ad ambienti mutevoli e prendere decisioni in condizioni di incertezza.
Il capitolo consente a ricercatori e professionisti di sviluppare soluzioni di IA in grado di resistere alle complessità e alle incertezze degli ambienti del mondo reale. Fornisce approfondimenti sui principi di progettazione, sulle architetture e sugli algoritmi alla base di sistemi IA robusti e una guida pratica per l'implementazione e la convalida di questi sistemi.

20.14 Plasmare il Futuro dei Sistemi ML

Guardando al futuro, la traiettoria dei sistemi ML punta verso un cambiamento di paradigma da un approccio incentrato sul modello a uno più incentrato sui dati. Questo cambiamento riconosce che la qualità e la diversità dei dati sono fondamentali per sviluppare modelli di IA solidi, affidabili ed equi.
Prevediamo una crescente enfasi sulle tecniche di cura dei dati, etichettatura e aumento nei prossimi anni. Queste pratiche mirano a garantire che i modelli siano addestrati su dati rappresentativi di alta qualità che riflettano accuratamente le complessità e le sfumature degli scenari del mondo reale. Concentrandoci sulla qualità e sulla diversità dei dati, possiamo mitigare i rischi di modelli parziali o distorti che possono perpetuare risultati ingiusti o discriminatori.
Questo approccio incentrato sui dati sarà fondamentale per affrontare le sfide di pregiudizio, equità e generalizzabilità nei sistemi ML. Cercando e incorporando attivamente set di dati diversi e inclusivi, possiamo sviluppare modelli più solidi, equi e applicabili per vari contesti e popolazioni. Inoltre, l'enfasi sui dati guiderà i progressi in tecniche come il "data augmentation", in cui i set di dati esistenti vengono ampliati e diversificati tramite sintesi, traduzione e generazione di dati. Queste tecniche possono aiutare a superare i limiti dei set di dati piccoli o sbilanciati, consentendo lo sviluppo di modelli più accurati e generalizzabili.
Negli ultimi anni, l'IA generativa ha preso d'assalto il campo, dimostrando notevoli capacità nella creazione di immagini, video e testo realistici. Tuttavia, l'ascesa dell'IA generativa porta anche nuove sfide per i sistemi ML (Capitolo 18). A differenza dei tradizionali sistemi ML, i modelli generativi spesso richiedono più risorse computazionali e pongono sfide in termini di scalabilità ed efficienza. Inoltre, la valutazione e il benchmarking dei modelli generativi presentano difficoltà, poiché le metriche tradizionali utilizzate per le attività di classificazione potrebbero non essere direttamente applicabili. Lo sviluppo di solidi framework di valutazione per i modelli generativi è un'area di ricerca attiva.
Comprendere e affrontare queste sfide di sistema e considerazioni etiche sarà fondamentale per dare forma al futuro dell'IA generativa e al suo impatto sulla società. In qualità di professionisti e ricercatori di ML, siamo responsabili dello sviluppo delle capacità tecniche dei modelli generativi e dello sviluppo di sistemi e framework solidi in grado di mitigare i potenziali rischi e garantire l'applicazione vantaggiosa di questa potente tecnologia.

20.15 Applicazione di "AI for Good"

Il potenziale dell'IA per essere utilizzata per il bene sociale è vasto, a condizione che vengano sviluppati e distribuiti sistemi ML responsabili su larga scala in vari casi d'uso (Capitolo 19). Per realizzare questo potenziale, è essenziale che ricercatori e professionisti si impegnino attivamente nel processo di apprendimento, sperimentazione e superamento dei limiti di ciò che è possibile.

Durante lo sviluppo dei sistemi ML, è fondamentale ricordare i temi e le lezioni chiave esplorati in questo libro. Questi includono l'importanza della qualità e della diversità dei dati, la ricerca di efficienza e robustezza, il potenziale di TinyML e del calcolo neuromorfico e l'imperativo della sicurezza e della privacy. Queste intuizioni informano il lavoro e guidano le decisioni di coloro che sono coinvolti nello sviluppo di sistemi di IA.

È importante riconoscere che lo sviluppo dell'IA non è solo un'impresa tecnica, ma anche profondamente umana. Richiede collaborazione, empatia e un impegno per comprendere le implicazioni sociali dei sistemi creati. Interagire con esperti di diversi campi, come etica, scienze sociali e politica, è essenziale per garantire che i sistemi di IA sviluppati siano tecnicamente validi, socialmente responsabili e utili. Cogliere l'opportunità di far parte di questo campo trasformativo e plasmarne il futuro è un privilegio e una responsabilità. Lavorando insieme, possiamo creare un mondo in cui i sistemi di ML fungono da strumenti per un cambiamento positivo e per migliorare la condizione umana.

20.16 Congratulazioni

Congratulazioni per essere arrivati fin qui e buona fortuna per gli sforzi futuri! Il futuro dell'intelligenza artificiale è luminoso e pieno di infinite possibilità. Sarà emozionante vedere gli incredibili contributi che darete in questo campo.

Sentitevi liberi di contattarmi in qualsiasi momento all'indirizzo vj at eecs dot harvard dot edu.

– *Prof. Vijay Janapa Reddi, Harvard University*

21. LABORATORI

Panoramica

Benvenuti alla sezione dei laboratori pratici in cui esplorerete l'implementazione di modelli ML su dispositivi embedded reali, che offriranno un'introduzione pratica ai sistemi ML. A differenza degli approcci tradizionali con modelli su larga scala, questi laboratori si concentrano sull'interazione diretta sia con l'hardware che con il software. Ci aiutano a mostrare varie modalità di sensori in diversi casi d'uso delle applicazioni. Questo approccio fornisce preziose informazioni sulle sfide e le opportunità dell'implementazione dell'IA su sistemi fisici reali.

Obiettivi dell'Apprendimento

Completando questi "lab", speriamo che gli studenti:

Consiglio

- Acquisiscano competenza nell'impostazione e distribuzione di modelli ML su dispositivi supportati, consentendo di affrontare scenari di distribuzione ML nel mondo reale con sicurezza.
- Comprendano i passaggi coinvolti nell'adattamento e nella sperimentazione di modelli ML per diverse applicazioni, consentendo di ottimizzare prestazioni ed efficienza.
- Apprendano tecniche di risoluzione dei problemi specifiche per le distribuzioni ML embedded, dotandoli delle competenze per superare insidie e sfide comuni.
- Acquisiscano esperienza pratica nella distribuzione di modelli TinyML su dispositivi embedded, colmando il divario tra teoria e pratica.
- Esplorino varie modalità di sensori e le loro applicazioni, ampliando la comprensione di come ML può essere sfruttato in diversi domini.
- Favoriscano una comprensione delle implicazioni e delle sfide del mondo reale associate alle distribuzioni di sistemi ML, preparandoli per progetti futuri.

Pubblico Target

Questi lab sono progettati per:

- **Principianti** nel campo dell'apprendimento automatico che hanno un vivo interesse nell'esplorare l'intersezione tra ML e sistemi embedded.
- **Sviluppatori e ingegneri** che desiderano applicare modelli ML ad applicazioni del mondo reale utilizzando dispositivi a basso consumo e risorse limitate.
- **Appassionati e ricercatori** che desiderano acquisire esperienza pratica nell'implementazione dell'IA su dispositivi edge e comprendere le sfide uniche coinvolte.

Dispositivi Supportati

Abbiamo incluso materiali di laboratorio per tre dispositivi chiave che rappresentano diversi profili hardware e capacità.

- Nicla Vision: Ottimizzato per applicazioni basate sulla visione come la classificazione delle immagini e il rilevamento di oggetti, ideale per casi d'uso compatti e a basso consumo.
- XIAO ESP32S3: Una scheda versatile e compatta adatta per attività di individuazione di parole chiave e rilevamento del movimento.

- Raspberry Pi: Una piattaforma flessibile per attività più intensive dal punto di vista computazionale, inclusi piccoli modelli linguistici e varie applicazioni di classificazione e rilevamento.

Esercizio	Nicla Vision	XIAO ESP32S3	Raspberry Pi
Installation & Setup	✓	✓	✓
Keyword Spotting (KWS)	✓	✓	
Image Classification	✓	✓	✓
Object Detection	✓	✓	✓
Motion Detection	✓	✓	
Small Language Models (SLM)			✓

Struttura del Lab

Ogni lab segue un approccio strutturato:

1. **Introduzione**: Esplora l'applicazione e la sua importanza in scenari reali.
2. **Configurazione**: Istruzioni dettagliate per configurare l'ambiente hardware e software.
3. **Distribuzione**: Guida al training e alla distribuzione dei modelli ML pre-addestrati sui dispositivi supportati.
4. **Esercizi**: Attività pratiche per modificare e sperimentare con i parametri del modello.
5. **Discussione**: Analisi dei risultati, potenziali miglioramenti e approfondimenti pratici.

Sequenza dei Laboratori Consigliata

Se si è alle prime armi con l'ML embedded, consigliamo di iniziare con la configurazione e l'individuazione delle parole chiave prima di passare alla classificazione delle immagini e al rilevamento degli oggetti. Gli utenti di Raspberry Pi possono esplorare attività più avanzate, come piccoli modelli linguistici, dopo aver familiarizzato con le basi.

Risoluzione dei Problemi e Supporto

Se si riscontrano problemi durante i laboratori, consultare i commenti sui "troubleshooting" [risoluzione dei problemi] o controllare le FAQ all'interno di ogni laboratorio. Per ulteriore assistenza, non esitate a contattare il nostro team di supporto o a interagire con i forum della community.

Crediti

Un ringraziamento speciale e un ringraziamento al Prof. Marcelo Rovai per il suo prezioso contributo allo sviluppo e al continuo perfezionamento di questi laboratori.

Guida Introduttiva

Benvenuti nell'entusiasmante mondo dell'apprendimento automatico embedded e di TinyML! In questa serie di laboratori pratici, esploreremo vari progetti che dimostrano la potenza dell'esecuzione di modelli di apprendimento automatico su dispositivi con risorse limitate. Prima di immergersi nei progetti, ci si assicuri di avere l'hardware e il software necessari.

Requisiti Hardware

Per seguire i laboratori pratici, ci sarà bisogno del seguente hardware:

1. **Scheda Arduino Nicla Vision**
    - Arduino Nicla Vision è una scheda potente e compatta progettata per applicazioni audio e di visione artificiale di livello professionale. È dotata di un modulo telecamera di alta qualità, un microfono digitale e un'IMU, che la rendono adatta a progetti impegnativi in settori quali robotica, automazione e sorveglianza.
    - Specifiche di Arduino Nicla Vision
    - Schema e pinout di Arduino Nicla Vision
2. **Scheda XIAO ESP32S3 Sense**
    - La scheda Seeed Studio XIAO ESP32S3 Sense è una scheda minuscola e ricca di funzionalità, progettata per maker, hobbisti e studenti interessati a esplorare applicazioni IA edge. È dotata di fotocamera, microfono e IMU, rendendo facile iniziare con progetti come classificazione delle immagini, individuazione di parole chiave e rilevamento del movimento.
    - Specifiche di XIAO ESP32S3 Sense
    - Schema e pinout di XIAO ESP32S3 Sense
3. **Raspberry Pi Single Computer board**

- Il Raspberry Pi è un potente e versatile computer a scheda singola che è diventato uno strumento essenziale per gli ingegneri di varie discipline. Sviluppati dalla Raspberry Pi Foundation, questi dispositivi compatti offrono una combinazione unica di convenienza, potenza di calcolo e ampie capacità GPIO (General Purpose Input/Output), rendendoli ideali per la prototipazione, lo sviluppo di sistemi embedded e progetti di ingegneria avanzata.
- Documentazione Hardware di Raspberry Pi
- Documentazione della Telecamera

4. **Accessori aggiuntivi**
    - Cavo USB-C per la programmazione e l'alimentazione di XIAO
    - Cavo micro-USB per la programmazione e l'alimentazione di Nicla
    - Alimentatore per Raspberry
    - Breadboard e cavi jumper (opzionali, per collegare sensori aggiuntivi)

Arduino Nicla Vision è progettata su misura per applicazioni di livello professionale, offrendo funzionalità avanzate e prestazioni adatte a progetti industriali impegnativi. D'altro canto, Seeed Studio XIAO ESP32S3 Sense è rivolta a maker, hobbisti e studenti che desiderano esplorare applicazioni IA edge in un formato più accessibile e adatto ai principianti. Entrambe le schede hanno i loro punti di forza e il loro pubblico di riferimento, consentendo agli utenti di scegliere la soluzione migliore per le loro esigenze e il loro livello di competenza. Raspberry Pi è destinato a progetti di ingegneria e apprendimento automatico più avanzati.

Requisiti Software

Per programmare le schede e sviluppare progetti di apprendimento automatico embedded, ci sarà bisogno del seguente software:

1. **Arduino IDE**
   - Download e installazione
     - Install di Arduino IDE
     - Seguire la guida all'installazione per il sistema operativo in uso.
     - Arduino CLI
     - Configurare l'IDE Arduino per le schede Arduino Nicla Vision e XIAO ESP32S3 Sense.
2. **OpenMV IDE (opzionale)**
   - Scaricare e installare OpenMV IDE per il sistema operativo in uso.
   - Configurare l'IDE OpenMV per Arduino Nicla Vision.
3. **Edge Impulse Studio**
   - Registrarsi per un account gratuito su Edge Impulse Studio.
   - Installare Edge Impulse CLI
   - Seguire le guide per connettere le schede Arduino Nicla Vision e XIAO ESP32S3 Sense a Edge Impulse Studio.
4. **Raspberry Pi OS**

- Scaricare e installare Raspberry Pi Imager

Connettività di Rete

Alcuni progetti potrebbero richiedere la connettività Internet per la raccolta dati o la distribuzione del modello. Assicurarsi che la connessione dell'ambiente di sviluppo sia stabile tramite Wi-Fi o Ethernet. Per Raspberry Pi, è necessaria una connessione Wi-Fi o Ethernet per il funzionamento remoto senza la necessità di collegare un monitor, una tastiera e un mouse.

- Per Arduino Nicla Vision, si può utilizzare il modulo Wi-Fi integrato per connetterti a una rete wireless.
- Per XIAO ESP32S3 Sense, si può utilizzare il modulo Wi-Fi integrato o collegare un modulo Wi-Fi o Ethernet esterno utilizzando i pin disponibili.
- Per Raspberry Pi, si può utilizzare il modulo Wi-Fi integrato per collegare un modulo Wi-Fi o Ethernet esterno utilizzando il connettore disponibile.

Conclusione

Con l'hardware e il software impostati, siamo pronti per intraprendere il viaggio di apprendimento automatico embedded. I laboratori pratici guideranno attraverso vari progetti, coprendo argomenti come classificazione delle immagini, rilevamento di oggetti, individuazione di parole chiave e classificazione del movimento.
Se si riscontrano problemi o ci sono domande, non esitate a consultare le guide alla risoluzione dei problemi o i forum o a cercare supporto dalla community.
Tuffiamoci e sblocchiamo il potenziale dell'apprendimento automatico su (tiny) sistemi reali!

Nicla Vision

Questi laboratori offrono un'opportunità unica per acquisire esperienza pratica con i sistemi di apprendimento automatico (ML). A differenza del lavoro con modelli di grandi dimensioni che richiedono risorse su scala di data center, questi esercizi consentono di interagire direttamente con hardware e software utilizzando TinyML. Questo approccio pratico offre una comprensione tangibile delle sfide e delle opportunità nell'implementazione dell'IA, sebbene su piccola scala. Tuttavia, i principi sono in gran parte gli stessi che si incontrerebbero quando si lavora con sistemi più grandi.

Nicla Vision. Fonte: Arduino

21.1 Prerequisiti

- **Nicla Vision Board**: Si deve avere la scheda Nicla Vision.
- **Cavo USB**: Per collegare la scheda al computer.
- **Rete**: Con accesso a Internet per scaricare il software necessario.

21.2 Setup

- Setup di Nicla Vision

21.3 Esercizi

Modalità	Attività	Descrizione	Link
Visione	Classificazione delle Immagini	Imparare a classificare le immagini	Link
Visione	Rilevamento degli Oggetti	Implementare il rilevamento degli oggetti	Link
Suono	Individuazione delle Parole Chiave	Esplorare i sistemi di riconoscimento vocale	Link
IMU	Classificazione del Movimento e Rilevamento delle Anomalie	Classifica i Dati di Movimento e Rileva le Anomalie	Link

Setup

DALL·E 3 Prompt: Illustrazione che ricorda un cartone animato degli anni '50 in cui la scheda Arduino NICLA VISION, dotata di vari sensori tra cui una telecamera, è il punto focale di una scrivania in stile antico. Sullo sfondo, uno schermo di computer con bordi arrotondati visualizza l'IDE Arduino. Il codice visualizzato è correlato alle configurazioni LED e al rilevamento dei comandi vocali tramite apprendimento automatico. Gli output sul monitor seriale visualizzano esplicitamente le parole 'yes' e 'no'.

Introduzione

La Arduino Nicla Vision (a volte chiamata *NiclaV*) è una scheda di sviluppo che include due processori in grado di eseguire attività in parallelo. Fa parte di una famiglia di schede di sviluppo con lo stesso fattore di forma ma progettate per attività specifiche, come Nicla Sense ME e la Nicla Voice. Le *Nicla* possono eseguire in modo efficiente processi creati con TensorFlow Lite. Ad esempio, uno dei core di NiclaV esegue un algoritmo di visione artificiale al volo (inferenza), mentre l'altro esegue operazioni di basso livello come il controllo di un motore e la comunicazione o l'azione come interfaccia utente. Il modulo wireless integrato consente la gestione simultanea della connettività WiFi e Bluetooth Low Energy (BLE).

Hardware

Due Core Paralleli

Il processore centrale è il dual-core STM32H747, che include un Cortex M7 at 480 MHz e un Cortex M4 at 240 MHz. I due core comunicano tramite un meccanismo di Remote Procedure Call che consente di richiamare senza problemi le funzioni sull'altro processore. Entrambi i processori condividono tutte le periferiche on-chip e possono eseguire:

- Sketch Arduino su Arm Mbed OS
- Applicazioni Native Mbed
- MicroPython / JavaScript tramite un interprete
- TensorFlow Lite

Memoria

La memoria è fondamentale per i progetti di machine learning [apprendimento automatico] embedded. La scheda NiclaV può ospitare fino a 16 MB di QSPI Flash per l'archiviazione. Tuttavia, è essenziale considerare che la SRAM MCU è quella da utilizzare con le inferenze di machine learning; l'STM32H747 è di soli 1 MB, condiviso da entrambi i processori. Questa MCU ha anche incluso 2 MB di FLASH, principalmente per l'archiviazione del codice.

Sensori

- **Fotocamera:** Una fotocamera CMOS a colori GC2145 da 2 MP.
- **Microfono:** l'`MP34DT05` è un microfono digitale MEMS omnidirezionale, ultracompatto, a basso consumo, costruito con un elemento di rilevamento capacitivo e l'interfaccia IC.
- **IMU a 6 Assi:** Dati del giroscopio 3D e dell'accelerometro 3D dall'IMU a 6 assi `LSM6DSOX`.
- **Sensore del Time of Flight:** Il sensore del tempo di volo `VL53L1CBV0FY` aggiunge capacità di misurazione precise e a bassa potenza alla Nicla Vision. Il laser invisibile VCSEL vicino all'infrarosso (incluso il driver analogico) è incapsulato con ottica ricevente in un piccolo modulo, tutto in uno, sotto la telecamera.

Installazione di Arduino IDE

Si inizia collegando la scheda (*microUSB*) al computer:

Si installa il core Mbed OS per le schede Nicla nell'IDE Arduino. Con l'IDE aperto, si va su `Tools > Board > Board Manager`, si cerca Arduino Nicla Vision nella finestra di ricerca e si installa la scheda.

Arduino IDE 1.18

Arduino IDE 2.X

Poi, si va su `Tools > Board > Arduino Mbed OS Nicla Boards` e si seleziona `Arduino Nicla Vision`. Con la scheda collegata alla porta USB, si dovrebbe vedere "Nicla on Port" e selezionarla.

> Si apre lo sketch Blink su Examples/Basic ed lo si esegue usando il pulsante "IDE Upload". Si dovrebbe vedere il LED integrato (RGB verde) lampeggiare, il che significa che la scheda Nicla è installata correttamente e funzionante!

Test del Microfono

Su Arduino IDE, si va su `Examples > PDM > PDMSerialPlotter`, si apre e si esegue lo sketch. Si apre il Plotter e si guarda la rappresentazione audio dal microfono:

Variare la frequenza del suono generato e verificare che il microfono funzioni correttamente.

Test dell'IMU

Prima di testare l'IMU, sarà necessario installare la libreria LSM6DSOX. Per farlo, si vai su Library Manager e si cerca LSM6DSOX. Si installa la libreria fornita da Arduino:

Poi, si va su `Examples > Arduino_LSM6DSOX > SimpleAccelerometer` e si esegue il test dell'accelerometro (si può anche eseguire Gyro e temperatura della scheda):

Test del sensore ToF (Time of Flight) [tempo di volo]

Come fatto con l'IMU, è necessario installare la libreria ToF VL53L1X. Per farlo, si va su Library Manager e si cerca VL53L1X. Si installa la libreria fornita da Pololu:

Poi, si esegue lo sketch proximity_detection.ino:

Sul monitor seriale, si vedrà la distanza dalla telecamera di un oggetto di fronte ad essa (max 4 m).

Test della Fotocamera

Possiamo anche testare la fotocamera utilizzando, ad esempio, il codice fornito in `Examples > Camera > CameraCaptureRawBytes`. Non possiamo vedere l'immagine direttamente, ma è possibile ottenere i dati "crudi" dell'immagine generati dalla telecamera.
In ogni caso, il test migliore con la telecamera è vedere un'immagine dal vivo. Per questo, useremo un altro IDE, OpenMV.

Installazione di OpenMV IDE

OpenMV IDE è il principale ambiente di sviluppo integrato con le telecamere OpenMV come quella su Nicla Vision. È dotato di un potente editor di testo, terminale di debug e visualizzatore di frame buffer con visualizzazione di istogrammi. Utilizzeremo MicroPython per programmare la telecamera.
Si va alla pagina di OpenMV IDE, si scarica la versione corretta per il proprio sistema operativo e si seguono le istruzioni per l'installazione sul computer.

L'IDE dovrebbe aprirsi, con il codice helloworld_1.py predefinito nella sua "Code Area". In caso contrario, lo si può aprire da `Files > Examples > HelloWord > helloword.py`

Tutti i messaggi inviati tramite una connessione seriale (utilizzando print() o i messaggi di errore) verranno visualizzati sul **Serial Terminal** durante l'esecuzione. L'immagine catturata da una telecamera verrà visualizzata nell'area **Camera Viewer** (o Frame Buffer) e nell'area Histogram immediatamente sotto Camera Viewer.

> Prima di collegare la Nicla all'IDE OpenMV, si deve avere la versione più recente del bootloader. Si va all'IDE Arduino, si seleziona la scheda Nicla e si apre lo sketch su `Examples > STM_32H747_System STM32H747_manageBootloader`. Upload-are il codice sulla board. Il Serial Monitor vi guiderà.

Dopo aver aggiornato il bootloader, si mette la Nicla Vision in modalità bootloader premendo due volte il pulsante di reset sulla scheda. Il LED verde integrato inizierà a spegnersi e accendersi in dissolvenza [fading]. Ora si torna all'IDE OpenMV e si clicca sull'icona di connessione (Left ToolBar):

Un pop-up informerà che è stata rilevata una scheda in modalità DFU e chiederà come si desidera procedere. Per prima cosa, si seleziona `Install the latest release firmware (vX.Y.Z)`. Questa azione installerà l'ultimo firmware OpenMV su Nicla Vision.

Si può lasciare l'opzione `Erase internal file system` deselezionata e cliccare su `[OK]`.
Il LED verde di Nicla inizierà a lampeggiare mentre il firmware OpenMV viene caricato sulla scheda e si aprirà una finestra del terminale che mostrerà l'avanzamento del flashing.

Attendere che il LED verde smetta di lampeggiare in dissolvenza. Quando il processo termina, si vedrà un messaggio che dice "DFU firmware update complete!". Premere `[OK]`.

Quando Nicla Vison si collega alla Barra degli Strumenti, compare un pulsante verde per la riproduzione.

Notare che sul computer apparirà un'unità denominata "NO NAME":

Ogni volta che si preme il pulsante `[RESET]` sulla board, viene automaticamente eseguito lo script *main.py* che vi è memorizzato. Si può caricare il codice di main.py sull'IDE (`File > Open File...`).

Questo è il codice di "Blink", che conferma che l'HW è OK.

Per testare la telecamera, eseguiamo *helloword_1.py*. Per farlo, si seleziona lo script su `File > Examples > HelloWorld > helloword.py`,
Quando si clicca sul pulsante di riproduzione verde, lo script MicroPython (*hellowolrd.py*) nella "Code Area" verrà caricato ed eseguito su Nicla Vision. Sul "Camera Viewer", si vedrà lo streaming video. Il "Serial Monitor" ci mostrerà gli FPS (fotogrammi al secondo), che dovrebbero essere circa 14 fps.

Ecco lo script helloworld.py:

```
# Hello World Example 2
#
# Welcome to the OpenMV IDE! Click on the green run arrow button below to run the script!

import sensor, image, time

sensor.reset()                      # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565) # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)   # Set frame size to QVGA (320x240)
sensor.skip_frames(time = 2000)     # Wait for settings take effect.
clock = time.clock()                # Create a clock object to track the FPS.

while(True):
    clock.tick()                    # Update the FPS clock.
    img = sensor.snapshot()         # Take a picture and return the image.
    print(clock.fps())
```

In GitHub, si trovano gli script Python utilizzati qui.
Il codice può essere suddiviso in due parti:

- **Setup:** Dove le librerie vengono importate, inizializzate e le variabili vengono definite e inizializzate.
- **Loop:** (ciclo while) parte del codice che viene eseguita continuamente. L'immagine (la variabile *img*) viene catturata (un frame). Ognuno di questi frame può essere utilizzato per l'inferenza nelle Applicazioni di Apprendimento Automatico.

Per interrompere l'esecuzione del programma, premere il pulsante rosso `[X]`.

> Nota: OpenMV Cam funziona a circa la metà della velocità quando è connesso all'IDE. Gli FPS dovrebbero aumentare una volta disconnessi.

In GitHub, si trovano altri script Python. Provare a testare i sensori di bordo.

Collegamento di Nicla Vision a Edge Impulse Studio

Avremo bisogno di Edge Impulse Studio più avanti in altri esercizi. Edge Impulse è una piattaforma di sviluppo leader per il machine learning [apprendimento automatico] su dispositivi edge.

Edge Impulse supporta ufficialmente Nicla Vision. Quindi, per iniziare, si crea un nuovo progetto su Studio e vi si collega Nicla. Per farlo, si seguono i passaggi:

- Scaricare l'EI Firmware più aggiornato e decomprimerlo.
- Aprire il file zip sul computer e selezionare l'uploader corrispondente al sistema operativo:

- Mettere Nicla-Vision in "Boot Mode", premendo due volte il pulsante di reset.
- Eseguire il codice batch specifico per il proprio sistema operativo per caricare il binario *arduino-nicla-vision.bin* sulla board.

Andare sul progetto sullo Studio e nella scheda `Data Acquisition tab`, selezionare `WebUSB` (1). Si aprirà una finestra; scegliere l'opzione che mostra che `Nicla is paired` (2) e premere `[Connect]` (3).

Nella sezione *Collect Data* della scheda `Data Acquisition`, si possono scegliere quali dati del sensore raccogliere.

Per esempio. `IMU data`:

O Image (`Camera`):

E così via. Si può anche testare un sensore esterno collegato all'`ADC` (Nicla pin 0) e agli altri sensori a bordo, come il microfono e il ToF.

Espansione della Nicla Vision Board (opzionale)

Un ultimo elemento da esplorare è che a volte, durante la prototipazione, è essenziale sperimentare con sensori e dispositivi esterni, e un'eccellente espansione per Nicla è l'Arduino MKR Connector Carrier (Grove compatible).
Lo shield ha 14 connettori Grove: cinque ingressi analogici singoli (A0-A5), un ingresso analogico doppio (A5/A6), cinque I/O digitali singoli (D0-D4), un I/O digitale doppio (D5/D6), una I2C (TWI) e una UART (Seriale). Tutti i connettori sono compatibili con 5 V.

> Notare che tutti i 17 pin Nicla Vision saranno collegati agli Shield Grove, ma alcune connessioni Grove rimangono scollegate.

Questo shield è compatibile con MKR e può essere utilizzato con Nicla Vision e Portenta.

Ad esempio, supponiamo che in un progetto TinyML si desideri inviare risultati di inferenza utilizzando un dispositivo LoRaWAN e aggiungere informazioni sulla luminosità locale. Spesso, con operazioni offline, si consiglia un display locale a basso consumo come un OLED. Questa configurazione può essere visualizzata qui:

Il Grove Light Sensor è collegato a uno dei singoli pin analogici (A0/PC4), il LoRaWAN device alla UART e l'OLED al connettore della I2C.
I pin Nicla 3 (Tx) e 4 (Rx) sono collegati al connettore Serial Shield. La comunicazione UART è utilizzata con il dispositivo LoRaWan. Ecco un semplice codice per utilizzare l'UART:

```
# UART Test - By: marcelo_rovai - Sat Sep 23 2023

import time
from pyb import UART
from pyb import LED

redLED = LED(1) # built-in red LED

# Init UART object.
# Nicla Vision's UART (TX/RX pins) is on "LP1"
uart = UART("LP1", 9600)

while(True):
    uart.write("Hello World!\r\n")
    redLED.toggle()
    time.sleep_ms(1000)
```

Per verificare che l'UART funzioni, si deve, ad esempio, collegare un altro dispositivo come Arduino UNO, visualizzando "Hello Word" sul Serial Monitor. Ecco il codice.

Di seguito è riportato il *codice Hello World* da utilizzare con l'OLED I2C. Il driver MicroPython SSD1306 OLED (ssd1306.py), creato da Adafruit, dovrebbe essere caricato anche su Nicla (lo script ssd1306.py si trova su GitHub).

```
# Nicla_OLED_Hello_World - By: marcelo_rovai - Sat Sep 30 2023

#Save on device: MicroPython SSD1306 OLED driver, I2C and SPI interfaces created by Adafr
uit
import ssd1306

from machine import I2C
i2c = I2C(1)

oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

oled.text('Hello, World', 10, 10)
oled.show()
Infine, ecco uno script semplice per leggere il valore ADC sul pin "PC4" (pin A0 di Nicla
):
# Light Sensor (A0) - By: marcelo_rovai - Wed Oct 4 2023

import pyb
from time import sleep

adc = pyb.ADC(pyb.Pin("PC4"))      # create an analog object from a pin
val = adc.read()                   # read an analog value

while (True):

    val = adc.read()
    print ("Light={}".format (val))
    sleep (1)
```

L'ADC può essere utilizzato per altre variabili del sensore, come la Temperatura.

Notare che gli script sopra (scaricati da Github) presentano solo come collegare dispositivi esterni alla scheda Nicla Vision utilizzando MicroPython.

Conclusione

Arduino Nicla Vision è un eccellente *piccolo dispositivo* per usi industriali e professionali! Tuttavia, è potente, affidabile, a basso consumo e ha sensori adatti per le applicazioni di machine learning embedded più comuni come visione, movimento, "fusion" di sensori e suono.

Nel repository GitHub, si trova l'ultima versione di tutti i codici utilizzati o commentati in questo esercizio pratico.

Risorse

- Codici Micropython
- Codici Arduino

Classificazione delle Immagini

DALL·E 3 Prompt: Fumetto in stile anni '50 raffigurante un dispositivo elettronico compatto con un modulo telecamera posizionato su un tavolo di legno. Lo schermo mostra un robot blu da un lato e dei pappagallini verdi dall'altro. Le luci LED sul dispositivo indicano le classificazioni, mentre i personaggi in abiti retrò osservano con interesse.

Introduzione

Mentre iniziamo i nostri studi sul machine learning embedded o TinyML, è impossibile ignorare l'impatto trasformativo della Computer Vision (CV) e dell'Intelligenza Artificiale (IA) nelle nostre vite. Queste due discipline interconnesse ridefiniscono ciò che le macchine possono percepire e realizzare, dai veicoli autonomi e dalla robotica all'assistenza sanitaria e alla sorveglianza.

Sempre di più, ci troviamo di fronte a una rivoluzione dell'intelligenza artificiale (IA) in cui, come affermato da Gartner, **Edge AI** ha un potenziale di impatto molto elevato, ed **è ora**!

Nel "centro" del Radar c'è la *Edge Computer Vision*, e quando parliamo di Machine Learning (ML) applicato alla visione, la prima cosa che viene in mente è la **Classificazione delle immagini**, una specie di "Hello World" di ML!

Questo esercizio esplorerà un progetto di computer vision che utilizza Convolutional Neural Network (CNN) [Reti Neurali Convoluzionali] per la classificazione delle immagini in tempo reale. Sfruttando il robusto ecosistema di TensorFlow, implementeremo un modello MobileNet pre-addestrato e lo adatteremo per il "deployment" edge. L'attenzione sarà rivolta all'ottimizzazione del modello per un'esecuzione efficiente su hardware con risorse limitate senza sacrificare l'accuratezza.

Utilizzeremo tecniche come la quantizzazione e la potatura per ridurre il carico computazionale. Alla fine di questo tutorial, si avrà un prototipo funzionante in grado di classificare le immagini in tempo reale, il tutto in esecuzione su un sistema embedded a basso consumo basato sulla scheda Arduino Nicla Vision.

Visione Artificiale

In sostanza, la visione artificiale consente alle macchine di interpretare e prendere decisioni sulla base di dati visivi provenienti dal mondo esterno, imitando sostanzialmente la capacità del sistema ottico umano. Al contrario, l'intelligenza artificiale è un campo più ampio che comprende il machine learning [apprendimento automatico], elaborazione del linguaggio naturale e robotica, tra le altre tecnologie. Quando si introducono algoritmi di IA nei progetti di visione artificiale, si potenzia la capacità del sistema di comprendere, interpretare e reagire agli stimoli visivi.
Quando si parla di progetti di visione artificiale applicati a dispositivi embedded, le applicazioni più comuni che vengono in mente sono *Classificazione delle Immagini* e *Rilevamento degli Oggetti*.

Entrambi i modelli possono essere implementati su dispositivi minuscoli come Arduino Nicla Vision e utilizzati in progetti reali. In questo capitolo, parleremo della classificazione delle immagini.

Obiettivo del Progetto di Classificazione delle Immagini

Il primo passo in qualsiasi progetto ML è definire l'obiettivo. In questo caso, è rilevare e classificare due oggetti specifici presenti in un'immagine. Per questo progetto, utilizzeremo due piccoli giocattoli: un *robot* e un piccolo pappagallo brasiliano (chiamato *Periquito*). Inoltre, raccoglieremo immagini di un *background* in cui quei due oggetti sono assenti.

Raccolta Dati

Una volta definito l'obiettivo del progetto di Machine Learning, il passaggio successivo e più cruciale è la raccolta del set di dati. Si può utilizzare Edge Impulse Studio, l'IDE OpenMV che

abbiamo installato o persino il proprop telefono per l'acquisizione delle immagini. Qui, utilizzeremo l'IDE OpenMV per questo.

Raccolta del Dataset con OpenMV IDE

Per prima cosa, si crea sul computer una cartella in cui verranno salvati i dati, ad esempio "data". Quindi, su OpenMV IDE, si va in `Tools > Dataset Editor` e si seleziona `New Dataset` per avviare la raccolta di dati:

L'IDE chiederà di aprire il file in cui verranno salvati i dati e di scegliere la cartella "data" che è stata creata. Notare che appariranno nuove icone sul pannello di sinistra.

Utilizzando l'icona in alto (1), si inserisce il nome della prima classe, ad esempio "periquito":

Eseguendo `dataset_capture_script.py` e cliccando sull'icona della fotocamera (2), inizierà l'acquisizione delle immagini:

Ripetere la stessa procedura con le altre classi

Suggeriamo circa 60 immagini da ogni categoria. Provare ad acquisire con diverse angolazioni, sfondi e condizioni di luce.

Le immagini archiviate utilizzano una dimensione del fotogramma QVGA di 320x240 e RGB565 (formato pixel colore).
Dopo aver acquisito il dataset, si chiude il Tool Dataset Editor su `Tools > Dataset Editor`.
Sul computer, si finirà con un set di dati che contiene tre classi: *periquito*, *robot* e *background*.

Si deve tornare a *Edge Impulse Studio* e caricare il set di dati nel progetto.

Addestramento del modello con Edge Impulse Studio

Useremo Edge Impulse Studio per addestrare il nostro modello. Inserire le credenziali del proprio account e creare un nuovo progetto:

Qui si può clonare un progetto simile: NICLA-Vision_Image_Classification.

Il Dataset

Utilizzando EI Studio (o *Studio*), esamineremo quattro passaggi principali per avere il nostro modello pronto per l'uso sulla scheda Nicla Vision: Dataset, Impulse, Tests e Deploy (su Edge Device, in questo caso, la NiclaV).

Per quanto riguarda il Dataset, è essenziale sottolineare che il nostro Dataset originale, acquisito con OpenMV IDE, sarà suddiviso in *Training*, *Validation* e *Test*. Il Test Set sarà suddiviso dall'inizio e una parte sarà riservata per essere utilizzata solo nella fase di Test dopo l'addestramento. Il Validation Set sarà utilizzato durante l'addestramento.

Su Studio, nella scheda Data acquisition e nella sezione UPLOAD DATA, si caricano i file delle categorie scelte dal computer:

Lasciare in Studio la suddivisione del dataset originale in *train and test* e scegliere l'etichetta relativa a quei dati specifici:

Ripetere la procedura per tutte e tre le classi. Alla fine, si vedranno "dati grezzi" in Studio:

Studio consente di esplorare i dati, mostrando una vista completa di tutti i dati nel progetto. Si possono cancellare, ispezionare o modificare le etichette cliccando sui singoli elementi di dati. Nel nostro caso, un progetto molto semplice, i dati sembrano OK.

Impulse Design

In questa fase, dovremmo definire come:

- Pre-elaborare i nostri dati, il che consiste nel ridimensionare le singole immagini e determinare la `profondità di colore` da utilizzare (RGB o scala di grigi) e

- Specificare un modello, in questo caso, sarà il `Transfer Learning (Images)` per mettere a punto un modello di classificazione delle immagini MobileNet V2 pre-addestrato sui nostri dati. Questo metodo funziona bene anche con set di dati di immagini relativamente piccoli (circa 150 immagini nel nostro caso).

Transfer Learning con MobileNet offre un approccio semplificato all'addestramento del modello, che è particolarmente utile per ambienti con risorse limitate e progetti con dati etichettati limitati. MobileNet, noto per la sua architettura leggera, è un modello pre-addestrato che ha già appreso funzionalità preziose da un ampio set di dati (ImageNet).

Sfruttando queste funzionalità apprese, si può addestrare un nuovo modello per il compito specifico con meno dati e risorse computazionali e tuttavia ottenere una precisione competitiva.

Questo approccio riduce significativamente i tempi di addestramento e i costi computazionali, rendendolo ideale per la prototipazione rapida e l'implementazione su dispositivi embedded in cui l'efficienza è fondamentale.
Si va alla scheda Impulse Design e si crea l'*impulse*, definendo una dimensione dell'immagine di 96x96 e schiacciandola (forma quadrata, senza ritaglio). Si seleziona Image e i blocchi Transfer Learning. Si salva l'Impulse.

Pre-elaborazione delle Immagini

Tutte le immagini QVGA/RGB565 in ingresso verranno convertite in 27.640 feature (96x96x3).

Si preme [Save parameters] e Generate all features:

Progettazione del Modello

Nel 2007, Google ha introdotto MobileNetV1, una famiglia di reti neurali per la visione artificiale di uso generale progettate pensando ai dispositivi mobili per supportare la classificazione, il rilevamento e altro ancora. Le MobileNet sono modelli piccoli, a bassa latenza e a basso consumo, parametrizzati per soddisfare i vincoli di risorse di vari casi d'uso. Nel 2018, Google ha lanciato MobileNetV2: Inverted Residuals and Linear Bottlenecks. MobileNet V1 e MobileNet V2 mirano all'efficienza mobile e alle applicazioni di visione embedded, ma differiscono per complessità architettonica e prestazioni. Mentre entrambi utilizzano convoluzioni separabili in profondità per ridurre i costi computazionali, MobileNet V2 introduce Inverted Residual Blocks e Linear Bottlenecks per migliorare le prestazioni. Queste nuove funzionalità consentono a V2 di acquisire funzionalità più complesse utilizzando meno parametri, rendendolo più efficiente dal punto di vista computazionale e generalmente più accurato rispetto al suo predecessore. Inoltre, V2 impiega un'attivazione non lineare nello layer di espansione intermedio. Utilizza ancora un'attivazione lineare per il layer del bottleneck [collo di bottiglia], una scelta di progettazione che si è rivelata utile per preservare informazioni importanti attraverso la rete. MobileNet V2 offre un'architettura ottimizzata per una maggiore accuratezza ed efficienza e verrà utilizzata in questo progetto. Sebbene l'architettura di base di MobileNet sia già minuscola e abbia una bassa latenza, molte volte, un caso d'uso o un'applicazione specifica potrebbe richiedere che il modello sia ancora più piccolo e veloce. MobileNets introduce un parametro semplice α (alfa) chiamato moltiplicatore di larghezza per costruire questi modelli più piccoli e meno costosi dal punto di vista computazionale. Il ruolo del moltiplicatore di larghezza α è quello di assottigliare una rete in modo uniforme a ogni layer.

Edge Impulse Studio può utilizzare sia MobileNetV1 (immagini 96x96) che V2 (immagini 96x96 o 160x160), con diversi valori di **α** (da 0,05 a 1,0). Ad esempio, si otterrà la massima accuratezza con V2, immagini 160x160 e α=1,0. Naturalmente, c'è un compromesso. Maggiore è la precisione, più memoria (circa 1,3 MB di RAM e 2,6 MB di ROM) sarà necessaria

per eseguire il modello, il che implica una maggiore latenza. L'ingombro minore sarà ottenuto all'altro estremo con MobileNetV1 e α=0,10 (circa 53,2 K di RAM e 101 K di ROM).

Per questo progetto utilizzeremo **MobileNetV2 96x96 0.1**, con un costo di memoria stimato di 265,3 KB in RAM. Questo modello dovrebbe andare bene per Nicla Vision con 1 MB di SRAM. Nella scheda Transfer Learning, si seleziona questo modello:

Addestramento del Modello

Un'altra tecnica preziosa da utilizzare con il Deep Learning è la **Data Augmentation**. Il "data augmentation" è un metodo per migliorare l'accuratezza dei modelli di apprendimento automatico mediante la creazione di dati artificiali aggiuntivi. Un sistema di Data Augmentation apporta piccole modifiche casuali ai dati di training (ad esempio capovolgendo, ritagliando o ruotando le immagini).

Guardando “sotto il cofano”, qui si può vedere come Edge Impulse implementa una policy di Data Augmentation sui dati:

```
# Implements the data augmentation policy
def augment_image(image, label):
    # Flips the image randomly
    image = tf.image.random_flip_left_right(image)

    # Increase the image size, then randomly crop it down to
    # the original dimensions
    resize_factor = random.uniform(1, 1.2)
    new_height = math.floor(resize_factor * INPUT_SHAPE[0])
    new_width = math.floor(resize_factor * INPUT_SHAPE[1])
    image = tf.image.resize_with_crop_or_pad(image, new_height, new_width)
    image = tf.image.random_crop(image, size=INPUT_SHAPE)

    # Vary the brightness of the image
    image = tf.image.random_brightness(image, max_delta=0.2)

    return image, label
```

L’esposizione a queste variazioni durante l’addestramento può aiutare a impedire al modello di prendere scorciatoie “memorizzando” indizi superficiali nei dati di addestramento, il che significa che potrebbe riflettere meglio i pattern profondi in esame nel set di dati.
L’ultimo layer del nostro modello avrà 12 neuroni con un dropout del 15% per prevenire l’overfitting. Ecco il risultato del Training:

Il risultato è eccellente, con 77 ms di latenza, che dovrebbero tradursi in 13 fps (frame al secondo) durante l’inferenza.

Test del Modello

Ora, si dovrebbe mettere da parte il set di dati all'inizio del progetto ed eseguire il modello addestrato usandolo come input:

Il risultato è, ancora una volta, eccellente.

Distribuzione del modello

A questo punto, possiamo distribuire il modello addestrato come .tflite e usare l'IDE OpenMV per eseguirlo usando MicroPython, oppure possiamo distribuirlo come C/C++ o libreria Arduino.

Libreria Arduino

Per prima cosa, distribuiamolo come Libreria Arduino:

Si dovrebbe installare la libreria come .zip sull'IDE Arduino ed eseguire lo sketch *nicla_vision_camera.ino* disponibile in Examples sotto il nome della libreria.

> Notare che Arduino Nicla Vision ha, per default, 512KB di RAM allocati per il core M7 e altri 244 KB sullo spazio di indirizzamento dell'M4. Nel codice, questa allocazione è stata modificata in 288 kB per garantire che il modello verrà eseguito sul dispositivo (`malloc_addblock((void*)0x30000000, 288 * 1024);`).

Il risultato è buono, con 86 ms di latenza misurata.

Ecco un breve video che mostra i risultati dell'inferenza: https://youtu.be/bZPZZJblU-o

OpenMV

È possibile distribuire il modello addestrato da utilizzare con OpenMV in due modi: come libreria e come firmware.

Come libreria vengono generati tre file: il modello .tflite addestrato, un elenco con etichette e un semplice script MicroPython che può effettuare inferenze utilizzando il modello.

Name	Size	Kind	Date Added
ei-nicla-vision_image_classification-openmv-v17	--	Folder	Today 14:59
trained.tflite	234 KB	TensorFlo...Lite Model	Today 14:59
labels.txt	26 bytes	Plain Text Document	Today 14:59
ei_image_classification.py	2 KB	Python File	Today 14:59
ei-nicla-vision_image_classification-openmv-v17.zip	140 KB	ZIP archive	Today 14:59

Eseguire questo modello come *.tflite* direttamente in Nicla era impossibile. Quindi, possiamo sacrificare l'accuratezza utilizzando un modello più piccolo o distribuire il modello come Firmware OpenMV (FW). Scegliendo FW, Edge Impulse Studio genera modelli, librerie e framework ottimizzati necessari per effettuare l'inferenza. Esploriamo questa opzione. Selezionare `OpenMV Firmware` nella scheda `Deploy` e premere `[Build]`.

Sul computer si troverà un file ZIP. Lo si apre:

Si usa il tool Bootloader sull'IDE OpenMV per caricare il FW sulla board:

Si seleziona il file appropriato (.bin per Nicla-Vision):

Dopo aver completato il download, si preme OK:

Se un messaggio dice che il FW è “outdated” [obsoleto], NON ESEGUIRE L’AGGIORNAMENTO. Selezionare [NO].

Ora, si apre lo script **ei_image_classification.py** che è stato scaricato da Studio e il file .bin per Nicla.

Eseguirlo. Puntando la telecamera sugli oggetti che vogliamo classificare, il risultato dell’inferenza verrà visualizzato sul Serial Terminal.

Modifica del Codice per Aggiungere Etichette

Il codice fornito da Edge Impulse può essere modificato in modo da poter vedere, per motivi di test, il risultato dell'inferenza direttamente sull'immagine visualizzata sull'IDE OpenMV. Caricare il codice da GitHub o modificalo come di seguito:

```
# Marcelo Rovai - NICLA Vision - Image Classification
# Adapted from Edge Impulse - OpenMV Image Classification Example
# @24Aug23

import sensor, image, time, os, tf, uos, gc

sensor.reset()                          # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)     # Set pxl fmt to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)       # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240))        # Set 240x240 window.
sensor.skip_frames(time=2000)           # Let the camera adjust.

net = None
labels = None

try:
    # Load built in model
    labels, net = tf.load_builtin_model('trained')
except Exception as e:
    raise Exception(e)

clock = time.clock()
while(True):
    clock.tick()  # Starts tracking elapsed time.

    img = sensor.snapshot()

    # default settings just do one detection
    for obj in net.classify(img,
                            min_scale=1.0,
                            scale_mul=0.8,
                            x_overlap=0.5,
                            y_overlap=0.5):
        fps = clock.fps()
        lat = clock.avg()

        print("**********\nPrediction:")
        img.draw_rectangle(obj.rect())
        # This combines the labels and confidence values into a list of tuples
```

```
        predictions_list = list(zip(labels, obj.output()))

        max_val = predictions_list[0][1]
        max_lbl = 'background'
        for i in range(len(predictions_list)):
            val = predictions_list[i][1]
            lbl = predictions_list[i][0]

            if val > max_val:
                max_val = val
                max_lbl = lbl

    # Print label with the highest probability
    if max_val < 0.5:
        max_lbl = 'uncertain'
    print("{} with a prob of {:.2f}".format(max_lbl, max_val))
    print("FPS: {:.2f} fps ==> latency: {:.0f} ms".format(fps, lat))

    # Draw label with highest probability to image viewer
    img.draw_string(
        10, 10,
        max_lbl + "\n{:.2f}".format(max_val),
        mono_space = False,
        scale=2
        )
```

Qui si può vedere il risultato:

Notare che la latenza (136 ms) è quasi il doppio di quella che ottenuta direttamente con l'IDE Arduino. Questo perché stiamo usando l'IDE come interfaccia e anche il tempo di attesa per la fotocamera per essere pronta. Se avviamo il clock appena prima dell'inferenza:

```
56 while(True):
57
58     img = sensor.snapshot()
59
60     clock.tick()  # Starts tracking elapsed time.
61
62     # default settings just do one detection... change them to search the image...
63     for obj in net.classify(img, min_scale=1.0, scale_mul=0.8, x_overlap=0.5, y_overlap=0.5):
64         fps = clock.fps()
65         lat = clock.avg()
66
67         print("**********\nPrediction:")
68         img.draw_rectangle(obj.rect())
69         # This combines the labels and confidence values into a list of tuples
70         predictions_list = list(zip(labels, obj.output()))
```

La latenza scenderà a soli 71 ms.

NiclaV funziona a circa la metà della velocità quando è connesso all'IDE. Gli FPS dovrebbero aumentare una volta disconnessi.

Post-elaborazione con i LED

Quando lavoriamo con l'apprendimento automatico embedded, cerchiamo dispositivi che possano procedere continuamente con l'inferenza e il risultato, eseguendo un'azione direttamente sul mondo fisico e non visualizzando il risultato su un computer connesso. Per simulare ciò, accenderemo un LED diverso per ogni possibile risultato dell'inferenza.

Per ottenere ciò, dovremmo caricare il codice da GitHub o modificare l'ultimo codice per includere i LED:

```
# Marcelo Rovai - NICLA Vision - Image Classification with LEDs
# Adapted from Edge Impulse - OpenMV Image Classification Example
# @24Aug23

import sensor, image, time, os, tf, uos, gc, pyb

ledRed = pyb.LED(1)
ledGre = pyb.LED(2)
ledBlu = pyb.LED(3)

sensor.reset()                          # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)     # Set pixl fmt to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)       # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240))        # Set 240x240 window.
sensor.skip_frames(time=2000)           # Let the camera adjust.

net = None
labels = None

ledRed.off()
ledGre.off()
```

```
ledBlu.off()

try:
    # Load built in model
    labels, net = tf.load_builtin_model('trained')
except Exception as e:
    raise Exception(e)

clock = time.clock()


def setLEDs(max_lbl):

    if max_lbl == 'uncertain':
        ledRed.on()
        ledGre.off()
        ledBlu.off()

    if max_lbl == 'periquito':
        ledRed.off()
        ledGre.on()
        ledBlu.off()

    if max_lbl == 'robot':
        ledRed.off()
        ledGre.off()
        ledBlu.on()

    if max_lbl == 'background':
        ledRed.off()
        ledGre.off()
        ledBlu.off()


while(True):
    img = sensor.snapshot()
    clock.tick()  # Starts tracking elapsed time.

    # default settings just do one detection.
    for obj in net.classify(img,
                            min_scale=1.0,
                            scale_mul=0.8,
                            x_overlap=0.5,
                            y_overlap=0.5):
        fps = clock.fps()
        lat = clock.avg()

        print("**********\nPrediction:")
        img.draw_rectangle(obj.rect())
        # This combines the labels and confidence values into a list of tuples
        predictions_list = list(zip(labels, obj.output()))

        max_val = predictions_list[0][1]
        max_lbl = 'background'
        for i in range(len(predictions_list)):
            val = predictions_list[i][1]
            lbl = predictions_list[i][0]

            if val > max_val:
                max_val = val
                max_lbl = lbl

    # Print label and turn on LED with the highest probability
    if max_val < 0.8:
        max_lbl = 'uncertain'

    setLEDs(max_lbl)

    print("{} with a prob of {:.2f}".format(max_lbl, max_val))
    print("FPS: {:.2f} fps ==> latency: {:.0f} ms".format(fps, lat))

    # Draw label with highest probability to image viewer
```

```
    img.draw_string(
        10, 10,
        max_lbl + "\n{:.2f}".format(max_val),
        mono_space = False,
        scale=2
        )
```

Ora, ogni volta che una classe ottiene un risultato superiore a 0,8, il LED corrispondente si accenderà:

- Led Rosso On: incerto (nessuna classe supera 0,8)
- Led Green On: Periquito > 0.8
- Led Blue On: Robot > 0.8
- Tutti i LED spenti: Sfondo > 0.8

Ecco il risultato:

Più in dettaglio

Classificazione delle immagini Benchmark (non ufficiale)

Diverse schede di sviluppo possono essere utilizzate per l'apprendimento automatico embedded (TinyML) e le più comuni per le applicazioni di Computer Vision (consumo energetico basso) sono ESP32 CAM, Seeed XIAO ESP32S3 Sense, Arduino Nicla Vison e Arduino Portenta.

	ESP 32	Seeed XIAO Sense / ESP32S3	Arduino Pro
32Bits CPU	Xtensa LX6 Dual Core	Arm Cortex-M4F (BLE) Xtensa LX7 Dual Core	Dual Core Arm Cortex M7/M4
CLOCK	240MHz	64 / 240MHz	480/240MHz
RAM	520KB (part available)	256KB / 8MB	1MB
ROM	2MB	2MB / 8MB	2MB
Radio	BLE/WiFi	BLE / WiFi (ESP32S3)	BLE/WiFi
Sensors	Yes (CAM)	Yes (Sense)	Yes (Nicla)
Bat. Power Manag.	No	Yes	Yes
Price	$	$$	$$$$$

Cogliendo l'occasione, lo stesso modello addestrato è stato distribuito su ESP-CAM, XIAO e Portenta (in questo caso, il modello è stato addestrato di nuovo, utilizzando immagini in scala di grigi per essere compatibile con la sua fotocamera). Ecco il risultato, distribuendo i modelli come Libreria di Arduino:

Conclusione

Prima di finire, si tenga presente che la Computer Vision è più di una semplice classificazione delle immagini. Ad esempio, si possono sviluppare progetti Edge Machine Learning sulla visione in diverse aree, come:

- **Veicoli Autonomi:** Usa un gruppo di sensori, i dati lidar e gli algoritmi di visione artificiale per navigare e prendere decisioni.
- **Sanità:** Diagnosi automatizzata di malattie tramite analisi delle immagini di risonanza magnetica, raggi X e TAC

- **Vendita al Dettaglio:** Sistemi di pagamento automatizzati che identificano i prodotti mentre passano attraverso uno scanner.
- **Sicurezza e Sorveglianza:** Riconoscimento facciale, rilevamento di anomalie e tracciamento di oggetti in video in tempo reale.
- **Realtà Aumentata:** Rilevamento e classificazione di oggetti per sovrapporre informazioni digitali al mondo reale.
- **Automazione Industriale:** Ispezione visiva di prodotti, manutenzione predittiva e guida di robot e droni.
- **Agricoltura;** Monitoraggio delle colture basato su droni e raccolta automatizzata.
- **Elaborazione del Linguaggio Naturale:** Didascalie delle immagini e risposte visive alle domande.
- **Riconoscimento dei Gesti:** Per giochi, traduzione del linguaggio dei segni e interazione uomo-macchina.
- **Raccomandazione dei Contenuti:** Sistemi di raccomandazione basati sulle immagini nell'e-commerce.

Risorse

- Codici Micropython
- Dataset
- Progetto Edge Impulse

Rilevamento degli Oggetti

DALL·E 3 Prompt: umetto in stile anni '40 o '50 che mostra l'interno di un ampio magazzino industriale. Un nastro trasportatore è in primo piano, trasportando un mix di ruote e scatole giocattolo. Le ruote sono distinguibili per i loro cerchioni giallo brillante e pneumatici neri. Le scatole sono cubi bianchi dipinti con pattern alternati in bianco e nero. Alla fine del nastro trasportatore in movimento c'è un robot in stile retrò, dotato di strumenti e sensori, che classifica e conta diligentemente le ruote e le scatole in arrivo. L'estetica generale ricorda l'animazione di metà secolo con linee audaci e una tavolozza di colori classica.

Introduzione

Questo è il seguito di **CV su Nicla Vision**, che ora esplora l'**Object Detection** sui microcontrollori.

Object Detection e Image Classification

Il compito principale dei modelli di Image Classification classificazione delle immagini è quello di produrre un elenco delle categorie di oggetti più probabili presenti in un'immagine, ad esempio, per identificare un gatto soriano subito dopo cena:

```
[PREDICTION]:

1) [tabby] ==> Probability of 30%
2) [bow tie] ==> Probability of 11%
3) [Egyptian cat] ==> Probability of 18%
```

Ma cosa succede quando il gatto salta vicino al bicchiere di vino? Il modello riconosce ancora solo la categoria predominante nell'immagine, il gatto soriano:

```
[PREDICTION]:

1) [tabby] ==> Probability of 53%
2) [tiger cat] ==> Probability of 23%
3) [Egyptian cat] ==> Probability of 10%
```

E cosa succede se non c'è una categoria dominante nell'immagine?

```
[PREDICTION]        [Prob]

ashcan              : 27%
Egyptian cat        : 19%
hamper              : 13%
```

Il modello identifica l'immagine soprastante in modo completamente sbagliato come un "ashcan" [?pattumiera?], probabilmente a causa delle tonalità di colore.

> Il modello utilizzato in tutti gli esempi precedenti è *MobileNet*, addestrato con un ampio set di dati, la *ImageNet*.

Per risolvere questo problema, abbiamo bisogno di un altro tipo di modello, in cui non solo possono essere trovate **più categorie** (o etichette), ma anche **dove** si trovano gli oggetti in una determinata immagine.

Come possiamo immaginare, tali modelli sono molto più complicati e più grandi, ad esempio, **MobileNetV2 SSD FPN-Lite 320x320, addestrato con il set di dati COCO**. Questo modello di rilevamento degli oggetti pre-addestrato è progettato per individuare fino a 10 oggetti all'interno di un'immagine, generando un riquadro di delimitazione per ogni oggetto rilevato. L'immagine sottostante è il risultato di un tale modello in esecuzione su un Raspberry Pi:

Quei modelli utilizzati per il rilevamento degli oggetti (come MobileNet SSD o YOLO) hanno solitamente dimensioni di diversi MB, il che è OK per l'uso con Raspberry Pi ma non adatto per l'uso con dispositivi embedded, dove la RAM solitamente è inferiore a 1 M Byte.

Una soluzione innovativa per il rilevamento degli oggetti: FOMO

Edge Impulse ha lanciato nel 2022 **FOMO** (Faster Objects, More Objects), una nuova soluzione per eseguire il rilevamento di oggetti su dispositivi embedded, non solo su Nicla Vision (Cortex M7) ma anche su CPU Cortex M4F (serie Arduino Nano33 e OpenMV M4) e sui dispositivi Espressif ESP32 (ESP-CAM e XIAO ESP32S3 Sense).
In questo esercizio pratico, esploreremo l'uso di FOMO con Object Detection, senza entrare in molti dettagli sul modello stesso. Per saperne di più su come funziona il modello, si può esaminare l'annuncio ufficiale FOMO di Edge Impulse, dove Louis Moreau e Mat Kelcey spiegano in dettaglio come funziona.

Obiettivo del Progetto di Object Detection

Tutti i progetti di apprendimento automatico devono iniziare con un obiettivo dettagliato. Supponiamo di trovarci in una struttura industriale e di dover ordinare e contare **ruote** e **scatole** speciali.

In altre parole, dovremmo eseguire una classificazione multi-etichetta, in cui ogni immagine può avere tre classi:

- Background [Sfondo] (nessun oggetto)
- Box [Scatola]
- Wheel [Ruota]

Ecco alcuni campioni di immagini non etichettate che dovremmo usare per rilevare gli oggetti (ruote e scatole):

Siamo interessati a quale oggetto è presente nell'immagine, alla sua posizione (centroide) e a quanti ne possiamo trovare su di essa. La dimensione dell'oggetto non viene rilevata con FOMO, come con MobileNet SSD o YOLO, in cui il Bounding Box è uno degli output del modello.

Svilupperemo il progetto utilizzando Nicla Vision per l'acquisizione di immagini e l'inferenza del modello. Il progetto ML verrà sviluppato utilizzando Edge Impulse Studio. Ma prima di iniziare il progetto di "object detection" in Studio, creiamo un *dataset grezzo* (non etichettato) con immagini che contengono gli oggetti da rilevare.

Raccolta Dati

Possiamo utilizzare Edge Impulse Studio, OpenMV IDE, il telefono o altri dispositivi per l'acquisizione delle immagini. Qui, utilizzeremo di nuovo OpenMV IDE per il nostro scopo.

Raccolta del Dataset con OpenMV IDE

Per prima cosa, si crea sul computer una cartella in cui verranno salvati i dati, ad esempio "data". Quindi, su OpenMV IDE, si va in "Tools > Dataset Editor" e si seleziona "New Dataset" per avviare la raccolta di dati:

Edge impulse suggerisce che gli oggetti dovrebbero essere di dimensioni simili e non sovrapposti per prestazioni migliori. Questo va bene in una struttura industriale, dove la telecamera dovrebbe essere fissa, mantenendo la stessa distanza dagli oggetti da rilevare. Nonostante ciò, proveremo anche con dimensioni e posizioni miste per vedere il risultato.

> Non creeremo cartelle separate per le nostre immagini perché ciascuna contiene più etichette.

Si collega Nicla Vision a OpenMV IDE e si esegue `dataset_capture_script.py`. Cliccando sul pulsante "Capture Image" inizierà l'acquisizione delle immagini:

Suggeriamo circa 50 immagini che mescolano gli oggetti e variano il numero di ciascuno che appare sulla scena. Provare ad acquisire con diverse angolazioni, sfondi e condizioni di luce.

> Le immagini memorizzate utilizzano una dimensione del fotogramma QVGA 320x240 e RGB565 (formato pixel a colori).

Dopo aver acquisito il dataset, si chiude il Tool Dataset Editor su `Tools > Dataset Editor`.

Edge Impulse Studio

Setup del progetto

Si va su Edge Impulse Studio, si inseriscono le proprie credenziali in **Login** (o si crea un account) e si avvia un nuovo progetto.

Qui si può clonare il progetto sviluppato per questa esercitazione pratica: NICLA_Vision_Object_Detection.

Nella "Project Dashboard", si va in basso e su **Project info** e si seleziona **Bounding boxes (object detection)** e Nicla Vision come "Target Device":

Caricamento dei dati non etichettati

Su Studio, si va alla scheda `Data acquisition` e nella sezione `UPLOAD DATA`, si caricano dal computer i file acquisiti.

Si può lasciare che Studio divida automaticamente i dati tra “Train” e “Test” o farlo manualmente.

Tutte le immagini non etichettate (51) sono state caricate, ma devono comunque essere etichettate in modo appropriato prima di utilizzarle come set di dati nel progetto. Lo Studio ha uno strumento per questo scopo, che si trova al link `Labeling queue (51)`.

Ci sono due modi per eseguire l’etichettatura assistita dall’IA su Edge Impulse Studio (versione gratuita):

- Utilizzando yolov5
- Tracciando di oggetti tra i frame

Edge Impulse ha lanciato una funzione di auto-labeling per i clienti Enterprise, semplificando le attività di etichettatura nei progetti di rilevamento degli oggetti.

Gli oggetti ordinari possono essere rapidamente identificati ed etichettati utilizzando una libreria esistente di modelli di rilevamento degli oggetti pre-addestrati da YOLOv5 (addestrati con il set di dati COCO). Ma poiché, nel nostro caso, gli oggetti non fanno parte dei set di dati COCO, dovremmo selezionare l'opzione di `tracking objects`. Con questa opzione, una volta disegnati i riquadri di delimitazione ed etichettate le immagini in un frame, gli oggetti verranno tracciati automaticamente da un frame all'altro, etichettando *parzialmente* quelli nuovi (non tutti sono etichettati correttamente).

Si può usare EI uploader per importare i dati se si ha già un dataset etichettato contenente dei "bounding box".

Etichettatura del Dataset

Iniziando dalla prima immagine dei dati non etichettati, si usa il mouse per trascinare una casella attorno a un oggetto per aggiungere un'etichetta. Poi si clicca su **Save labels** per passare all'elemento successivo.

Si continua con questo processo finché la coda non è vuota. Alla fine, tutte le immagini dovrebbero avere gli oggetti etichettati come i campioni sottostanti:

Poi, si esaminano i campioni etichettati nella scheda `Data acquisition`. Se un'etichetta è sbagliata, la si può modificare usando il menù *`three dots`* dopo il nome del campione:

Si verrà guidati a sostituire l'etichetta sbagliata, correggendo il dataset.

Impulse Design

In questa fase, si deve definire come:

- Il **Pre-processing** consiste nel ridimensionare le singole immagini da `320 x 240` a `96 x 96` e nel comprimerle (forma quadrata, senza ritaglio). Successivamente, le immagini vengono convertite da RGB a scala di grigi.
- **Design a Model**, in questo caso, "Object Detection".

Pre-elaborazione di tutti i dataset

In questa sezione, si seleziona **Color depth** come `Grayscale`, che è adatta per l'uso con modelli FOMO e Save `parameters`.

Lo Studio passa automaticamente alla sezione successiva, `Generate features`, dove tutti i campioni saranno pre-elaborati, con conseguente set di dati con singole immagini 96x96x1 o 9.216 "feature".

L'esploratore di feature mostra che tutti i campioni evidenziano una buona separazione dopo la generazione delle feature.

Uno dei campioni (46) apparentemente si trova nello spazio sbagliato, ma cliccandoci sopra si può confermare che l'etichettatura è corretta.

Progettazione, Addestramento e Test del Modello

Useremo FOMO, un modello di rilevamento degli oggetti basato su MobileNetV2 (alpha 0.35) progettato per segmentare grossolanamente un'immagine in una griglia di **background** rispetto a **oggetti di interesse** (in questo caso, *scatole* e *ruote*).
FOMO è un modello di apprendimento automatico innovativo per il rilevamento degli oggetti, che può utilizzare fino a 30 volte meno energia e memoria rispetto ai modelli tradizionali come Mobilenet SSD e YOLOv5. FOMO può funzionare su microcontrollori con meno di 200 KB di RAM. Il motivo principale per cui ciò è possibile è che mentre altri modelli calcolano le dimensioni dell'oggetto disegnando un quadrato attorno ad esso (bounding box), FOMO ignora le dimensioni dell'immagine, fornendo solo le informazioni su dove si trova l'oggetto nell'immagine, tramite le coordinate del centroide.

Come funziona FOMO?

FOMO prende l'immagine in scala di grigi e la divide in blocchi di pixel usando un fattore di 8. Per l'input di 96x96, la griglia è 12x12 (96/8=12). Successivamente, FOMO eseguirà un classificatore attraverso ogni blocco di pixel per calcolare la probabilità che ci sia una scatola o una ruota in ognuno di essi e, successivamente, determinerà le regioni che hanno la più alta probabilità di contenere l'oggetto (se un blocco di pixel non ha oggetti, verrà classificato come *background*). Dalla sovrapposizione della regione finale, FOMO fornisce le coordinate (relative alle dimensioni dell'immagine) del centroide di questa regione.

Per l’addestramento, dovremmo selezionare un modello pre-addestrato. Usiamo `FOMO (Faster Objects, More Objects) MobileNetV2 0.35`. Questo modello usa circa 250 KB di RAM e 80 KB di ROM (Flash), che si adatta bene alla nostra scheda poiché ha 1 MB di RAM e ROM.

Per quanto riguarda gli iperparametri di training, il modello verrà addestrato con:

- Epochs: 60,
- Batch size: 32
- Learning Rate: 0.001.

Per la convalida durante l'addestramento, il 20% del set di dati (*validation_dataset*) verrà risparmiato. Per il restante 80% (*train_dataset*), applicheremo il "Data Augmentation", che capovolgerà casualmente, cambierà le dimensioni e la luminosità dell'immagine e le ritaglierà, aumentando artificialmente il numero di campioni sul set di dati per l'addestramento.
Di conseguenza, il modello termina con praticamente 1,00 nel punteggio F1, con un risultato simile quando si utilizzano i dati di test.

Notare che FOMO ha aggiunto automaticamente una terza etichetta di background [sfondo] ai due precedentemente definiti (*box* e *wheel*).

Nelle attività di rilevamento degli oggetti, l'accuratezza non è generalmente la evaluation metric primaria. Il rilevamento degli oggetti comporta la classificazione degli oggetti e la fornitura di riquadri di delimitazione attorno a essi, il che lo rende un problema più complesso della semplice classificazione. Il problema è che non abbiamo il riquadro di delimitazione, solo i centroidi. In breve, usare l'accuratezza come metrica potrebbe essere fuorviante e potrebbe non fornire una comprensione completa delle prestazioni del modello. Per questo motivo, useremo il punteggio F1.

Modello di test con "Live Classification"

Dato che Edge Impulse supporta ufficialmente Nicla Vision, colleghiamolo allo Studio. Per farlo, si seguono i passaggi:

- Si effettua il download dell'last EI Firmware e lo si decomprime.
- Si apre il file zip sul computer e si seleziona l'uploader relativo al proprio sistema operativo:

- Mettere Nicla-Vision in "Boot Mode", premendo due volte il pulsante di reset.

- Eseguire il codice batch specifico per il sistema operativo per caricare il binario (`arduino-nicla-vision.bin`) sulla board.

Si va nella sezione `Live classification` su EI Studio e, tramite *webUSB*, si connette la Nicla Vision:

Una volta connessa, si può usare la Nicla per catturare immagini reali da testare col modello addestrato su Edge Impulse Studio.

Una cosa da notare è che il modello può produrre falsi positivi e falsi negativi. Questo può essere ridotto al minimo definendo una `Confidence Threshold` [Soglia di confidenza] (si usa il menù `Three dots` [tre-punti] per la configurazione). Provare con 0,8 o più.

Distribuzione del Modello

Selezionare OpenMV Firmware nella scheda Deploy e premere [Build].

Quando si connette di nuovo Nicla con OpenMV IDE, proverà ad aggiornarne il FW. Scegliere invece l'opzione `Load a specific firmware`.

Si troverà un file ZIP sul computer dallo Studio. Lo si apre:

Caricare il file .bin sulla board:

Una volta terminato il download, verrà visualizzato un messaggio pop-up. `Premere OK` e aprire lo script **ei_object_detection.py** scaricato da Studio.

Prima di eseguire lo script, modifichiamo alcune righe. Notare che si può lasciare la definizione della finestra come 240 x 240 e la telecamera che cattura le immagini come QVGA/RGB. L'immagine catturata verrà pre-elaborata dal FW distribuito da Edge Impulse

```
# Edge Impulse - OpenMV Object Detection Example

import sensor, image, time, os, tf, math, uos, gc

sensor.reset()                          # Reset and initialize the sensor.
sensor.set_pixformat(sensor.RGB565)     # Set pixel format to RGB565 (or GRAYSCALE)
sensor.set_framesize(sensor.QVGA)       # Set frame size to QVGA (320x240)
sensor.set_windowing((240, 240))        # Set 240x240 window.
sensor.skip_frames(time=2000)           # Let the camera adjust.

net = None
labels = None
```

Ridefinire la confidenza minima, ad esempio, a 0,8 per ridurre al minimo i falsi positivi e negativi.

```
min_confidence = 0.8
```

Se necessario, modificare il colore dei cerchi che saranno utilizzati per visualizzare il centroide dell'oggetto rilevato per un contrasto migliore.

```
try:
    # Load built in model
    labels, net = tf.load_builtin_model('trained')
except Exception as e:
    raise Exception(e)

colors = [ # Add more colors if you are detecting more than 7 types of classes at once.
    (255, 255,   0), # background: yellow (not used)
    (  0, 255,   0), # cube: green
    (255,   0,   0), # wheel: red
    (  0,   0, 255), # not used
    (255,   0, 255), # not used
    (  0, 255, 255), # not used
    (255, 255, 255), # not used
]
```

Mantenere il codice restante così com'è e premere il `pulsante Play verde` per eseguire il codice:

Nella vista della telecamera, possiamo vedere gli oggetti con i loro centroidi contrassegnati con 12 cerchi pixel-fixed (ogni cerchio ha un colore distinto, a seconda della sua classe). Sul terminale seriale, il modello mostra le etichette rilevate e la loro posizione sulla finestra dell'immagine (240X240).

Attenzione l'origine delle coordinate è nell'angolo in alto a sinistra.

Notare che la frequenza dei fotogrammi al secondo è di circa 8 fps (simile a quella ottenuta con il progetto Image Classification). Ciò accade perché FOMO è intelligentemente costruito su un modello CNN, non con un modello di rilevamento degli oggetti come SSD MobileNet. Ad esempio, quando si esegue un modello MobileNetV2 SSD FPN-Lite 320x320 su un Raspberry Pi 4, la latenza è circa 5 volte superiore (circa 1,5 fps)
Ecco un breve video che mostra i risultati dell'inferenza: https://youtu.be/JbpoqRp3BbM

Conclusione

FOMO è un salto significativo nello spazio di elaborazione delle immagini, come hanno affermato Louis Moreau e Mat Kelcey durante il suo lancio nel 2022:

> FOMO è un algoritmo rivoluzionario che porta per la prima volta il rilevamento, il tracciamento e il conteggio degli oggetti in tempo reale sui microcontrollori.

Esistono molteplici possibilità per esplorare il rilevamento degli oggetti (e, più precisamente, il loro conteggio) su dispositivi embedded, ad esempio, per esplorare la il raggruppamento di sensori (telecamera + microfono) e il rilevamento degli oggetti di Nicla. Questo può essere molto utile nei progetti che coinvolgono le api, ad esempio.

Risorse

- Progetto Edge Impulse

Keyword Spotting (KWS)

DALL·E 3 Prompt: Scena di un cartone animato in stile anni '50 ambientata in una sala di ricerca audio vintage. Al centro ci sono due scienziate afroamericane. Una tiene una lente di ingrandimento, esaminando attentamente circuiti antichi, mentre l'altra prende appunti. Sul loro tavolo di legno ci sono diverse schede con sensori, in particolare un microfono. Dietro queste schede, un computer con un ampio display arrotondato mostra l'IDE Arduino. L'IDE mostra il codice per le assegnazioni dei pin LED e l'inferenza di apprendimento automatico per il rilevamento dei comandi vocali. Una finestra distinta nell'IDE, il monitor seriale, rivela output che indicano i comandi vocali 'yes' e 'no'. L'atmosfera della stanza è nostalgica con lampade vintage, strumenti di analisi audio classici e grafici FFT e curve del dominio del tempo.

Introduzione

Dopo aver già esplorato la scheda Nicla Vision nelle applicazioni di *Image Classification* e *Object Detection*, stiamo ora spostando la nostra attenzione sulle applicazioni attivate tramite comando vocale con un progetto su Keyword Spotting (KWS).

Come introdotto nel tutorial *Feature Engineering for Audio Classification*, il Keyword Spotting (KWS) è integrato in molti sistemi di riconoscimento vocale, consentendo ai dispositivi di rispondere a parole o frasi specifiche. Sebbene questa tecnologia sia alla base di dispositivi popolari come Google Assistant o Amazon Alexa, è ugualmente applicabile e fattibile su dispositivi più piccoli e a basso consumo. Questo tutorial guiderà nell'implementazione di un

sistema KWS utilizzando TinyML sulla scheda di sviluppo Nicla Vision dotata di un microfono digitale.
Il nostro modello sarà progettato per riconoscere parole chiave che possono riattivare un dispositivo o azioni specifiche, dando loro vita con comandi vocali.

Come funziona un assistente vocale?

Come detto, gli *assistenti vocali* sul mercato, come Google Home o Amazon Echo-Dot, reagiscono agli umani solo quando vengono "svegliati" da parole chiave particolari come "Hey Google" sul primo e "Alexa" sul secondo.

In altre parole, il riconoscimento dei comandi vocali si basa su un modello multi-fase o Cascade Detection.

Fase 1: Un piccolo microprocessore all'interno di Echo Dot o Google Home ascolta continuamente, in attesa che venga individuata la parola chiave, utilizzando un modello TinyML nel device (applicazione KWS).

Fase 2: Solo quando vengono attivati dall'applicazione KWS nella Fase 1, i dati vengono inviati al cloud ed elaborati su un modello più grande.
Il video qui sotto mostra un esempio di un Google Assistant programmato su un Raspberry Pi (Fase 2), con un Arduino Nano 33 BLE come dispositivo TinyML (Fase 1).
https://youtu.be/e_OPgcnsyvM

> Per esplorare il progetto Google Assistant di cui sopra, consultare il tutorial: Building an Intelligent Voice Assistant From Scratch.

In questo progetto KWS, ci concentreremo sulla Fase 1 (KWS o Keyword Spotting), dove utilizzeremo Nicla Vision, che ha un microfono digitale che verrà utilizzato per individuare la parola chiave.

Il Progetto Pratico KWS

Il diagramma seguente fornisce un'idea di come dovrebbe funzionare l'applicazione KWS finale (durante l'inferenza):

La nostra applicazione KWS riconoscerà quattro classi di suono:

- **YES** (Keyword 1)
- **NO** (Keyword 2)
- **NOISE** (nessuna parola pronunciata; è presente solo rumore di fondo)
- **UNKNOW** (un mix di parole diverse da YES e NO)

Per progetti reali, è sempre consigliabile includere altri suoni oltre alle parole chiave, come "Noise" (o Background) e "Unknown".

Il Flusso di Lavoro del Machine Learning

Il componente principale dell'applicazione KWS è il suo modello. Quindi, dobbiamo addestrare un modello del genere con le nostre parole chiave specifiche, rumore e altre parole (lo "unknown"):

Il Dataset

Il componente critico di qualsiasi flusso di lavoro di apprendimento automatico è il **dataset**. Una volta decise le parole chiave specifiche, nel nostro caso (*YES* e NO), possiamo sfruttare il

dataset sviluppato da Pete Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". Questo dataset ha 35 parole chiave (con +1.000 campioni ciascuna), come yes, no, stop e go. In parole come *yes* e *no*, possiamo ottenere 1.500 campioni.
Si può scaricare una piccola parte del dataset da Edge Studio (Keyword spotting pre-built dataset), che include campioni dalle quattro classi che utilizzeremo in questo progetto: yes, no, noise e background. Per farlo, si seguono i passaggi seguenti:

- Download del dataset delle parole chiave.
- Unzip del file in una posizione a scelta.

Caricamento del set di dati su Edge Impulse Studio

Si avvia un nuovo progetto su Edge Impulse Studio (EIS) e si seleziona il tool `Upload Existing Data` nella sezione `Data Acquisition`. Si scelgono i file da caricare:

Si definisce l'etichetta, si seleziona `Automatically split between train and test`, e poi `Upload data` su EIS. Si ripete per tutte le classi.

Il dataset apparirà ora nella sezione `Data acquisition`. Notare che i circa 6.000 campioni (1.500 per ogni classe) sono suddivisi in set di Train (4.800) e di Test (1.200).

Acquisizione di Dati Audio Aggiuntivi

Sebbene disponiamo di molti dati dal dataset di Pete, è consigliabile raccogliere alcune parole pronunciate da noi. Lavorando con gli accelerometri, è essenziale creare un set di dati con dati acquisiti dallo stesso tipo di sensore. Nel caso del *suono*, questo è facoltativo perché ciò che classificheremo sono, in realtà, dati *audio*.

> La differenza fondamentale tra suono e audio è il tipo di energia. Il suono è una perturbazione meccanica (onde sonore longitudinali) che si propagano attraverso un mezzo, causando variazioni di pressione in esso. L'audio è un segnale elettrico (analogico o digitale) che rappresenta il suono.

Quando pronunciamo una parola chiave, le onde sonore devono essere convertite in dati audio. La conversione deve essere eseguita campionando il segnale generato dal microfono a una frequenza di 16 KHz con ampiezza di 16 bit per campione.
Quindi, qualsiasi dispositivo in grado di generare dati audio con questa specifica di base (16 KHz/16 bit) funzionerà correttamente. Come *device*, possiamo usare NiclaV, un computer o persino il cellulare.

Utilizzo di NiclaV ed Edge Impulse Studio

Come abbiamo appreso nel capitolo *Configurazione di Nicla Vision*, EIS supporta ufficialmente Nicla Vision, semplificando l'acquisizione dei dati dai suoi sensori, incluso il microfono. Quindi, si crea un nuovo progetto su EIS e vi si collega la Nicla, seguendo questi passaggi:

- Download del Firmware EIS più aggiornato e lo si decomprime.
- Aprire il file zip sul computer e selezionare l'uploader corrispondente al sistema operativo:

- Si mette NiclaV in Boot Mode premendo due volte il pulsante di reset.
- Si carica il binario *arduino-nicla-vision.bin* sulla board eseguendo il codice batch corrispondente al sistema operativo.

Si va sul proprio progetto EIS e nella scheda `Data Acquisition tab`, si seleziona `WebUSB`. Apparirà una finestra; scegliere l'opzione che mostra che `Nicla is paired` e si preme `[Connect]`.

Si possono scegliere quali dati del sensore raccogliere nella sezione `Collect Data` nella scheda `Data Acquisition`. Si seleziona: `Built-in microphone`, si definisce la `label` (per esempio, *yes*), la `Frequency`[16000Hz] e la `Sample length (in milliseconds)`, per esempio [10s]. `Start sampling`.

I dati sul dataset di Pete hanno una lunghezza di 1s, ma i campioni registrati sono lunghi 10s e devono essere suddivisi in campioni da 1s. Cliccare sui `tre puntini` dopo il nome del campione e selezionare `Split sample`.
Si aprirà una finestra col tool Split.

Una volta all'interno dello strumento, si dividono i dati in record da 1 secondo (1000 ms). Se necessario, si aggiungono o rimuovono segmenti. Questa procedura deve essere ripetuta per tutti i nuovi campioni.

Utilizzo di uno smartphone e di EI Studio

Si può anche utilizzare il PC o lo smartphone per acquisire dati audio, utilizzando una frequenza di campionamento di 16 KHz e una profondità di 16 bit.
Si va su `Devices`, si scansiona il `QR Code` col telefono e si clicca sul link. Un'app, Collection, per la raccolta dei dati apparirà nel browser. Si seleziona `Collecting Audio` e si definisce la `Label`, la `Length` [lunghezza] dei dati catturati e la `Category`.

Si ripete la stessa procedura usata con NiclaV.

Notare che qualsiasi app, come Audacity, può essere usata per la registrazione audio, a condizione che si usino campioni di 16KHz/16-bit.

Creazione di Impulse (Pre-Process / Definizione del Modello)

Un **impulse** *prende dati grezzi, usa l'elaborazione del segnale per estrarre le feature e poi usa un blocco di apprendimento per classificare nuovi dati.*

Impulse Design

Innanzitutto, prenderemo i dati con una finestra di 1 secondo, aumentando i dati e facendo scorrere quella finestra in intervalli di 500 ms. Notare che è impostata l'opzione Zero-pad data. È essenziale riempire i campioni con degli 'zeri' inferiori a 1 secondo (in alcuni casi, alcuni campioni possono risultare inferiori alla finestra di 1000 ms sul tool di suddivisione per evitare rumore e picchi).
Ogni campione audio di 1 secondo dovrebbe essere pre-elaborato e convertito in un'immagine (ad esempio, 13 x 49 x 1). Come discusso nel tutorial *Feature Engineering for Audio Classification*, utilizzeremo `Audio (MFCC)`, che estrae le feature dai segnali audio utilizzando i Mel Frequency Cepstral Coefficients, che sono adatti alla voce umana, il nostro caso qui.
Successivamente, selezioniamo il blocco `Classification` per costruire il nostro modello da zero utilizzando una Convolution Neural Network (CNN).

In alternativa, si può utilizzare il blocco `Transfer Learning (Keyword Spotting)`, che ottimizza un modello di keyword spotting pre-addestrato sui dati. Questo approccio ha buone prestazioni con set di dati di parole chiave relativamente piccoli.

Pre-elaborazione (MFCC)

Il passaggio successivo consiste nel creare le feature da addestrare nella fase successiva: Potremmo mantenere i valori di default dei parametri, ma utilizzeremo l'opzione DSP `Autotune parameters`.

Prenderemo le `Raw features` (i dati audio campionati a 16 KHz e 1 secondo) e utilizzeremo il blocco di elaborazione MFCC per calcolare le `Processed features`. Per ogni 16.000 feature grezze (16.000 x 1 secondo), otterremo 637 feature elaborate (13 x 49).

Il risultato mostra che abbiamo utilizzato solo una piccola quantità di memoria per pre-elaborare i dati (16 KB) e una latenza di 34 ms, il che è eccellente. Ad esempio, su un Arduino Nano (Cortex-M4f @ 64 MHz), lo stesso pre-processo richiederà circa 480 ms. I parametri scelti, come la `FFT length` [512], avranno un impatto significativo sulla latenza.
Ora, `Save parameters` e passiamo alla scheda `Generated features`, dove verranno generate le feature effettive. Utilizzando UMAP, una tecnica di riduzione delle dimensioni, `Feature explorer` mostra come le feature sono distribuite su un grafico bidimensionale.

Il risultato sembra OK, con una separazione visivamente netta tra feature *yes* (in rosso) e *no* (in blu). Le feature *unknown* [sconosciute] sembrano più vicine allo *spazio dei no* che a quello degli *yes*. Questo suggerisce che la parola chiave *no* ha una maggiore propensione ai falsi positivi.

Andiamo sotto il cofano

Per comprendere meglio come viene pre-elaborato il suono grezzo, leggere il capitolo *Feature Engineering for Audio Classification*. Si può sperimentare con la generazione di feature MFCC scaricando questo notebook da GitHub o [Opening it In Colab]

Progettazione e Addestramento del Modello

Useremo un semplice modello di rete neurale convoluzionale (CNN), testato con convoluzioni 1D e 2D. L'architettura di base ha due blocchi di Convolution + MaxPooling (filtri ([8] e [16], rispettivamente) e un Dropout di [0.25] per 1D e [0.5] per 2D. Per l'ultimo layer, dopo Flattening, abbiamo [4] neuroni, uno per ogni classe:

Come iperparametri, avremo un `Learning Rate` di [0.005] e un modello addestrato da [100] epoche. Includeremo anche un metodo di aumento dei dati basato su SpecAugment. Abbiamo addestrato i modelli 1D e 2D con gli stessi iperparametri. L'architettura 1D ha avuto

un risultato complessivo migliore (90,5% di accuratezza rispetto all'88% del 2D, quindi useremo l'1D.

ACCURACY
90.5%

LOSS
0.30

Confusion matrix (validation set) **1D Convolution**

	NO	NOISE	UNKNOWN	YES
NO	93.1%	0.8%	6.2%	0%
NOISE	0%	92.1%	7.5%	0.4%
UNKNOWN	10.5%	3.4%	84.9%	1.3%
YES	1.8%	1.4%	5.0%	91.9%
F1 SCORE	0.91	0.93	0.83	0.95

ACCURACY
88.0%

LOSS
0.34

Confusion matrix (validation set) **2D Convolution**

	NO	NOISE	UNKNOWN	YES
NO	88.4%	0.8%	9.3%	1.5%
NOISE	0%	96.0%	3.6%	0.4%
UNKNOWN	12.6%	6.3%	74.4%	6.7%
YES	1.8%	1.4%	3.6%	93.2%
F1 SCORE	0.88	0.94	0.78	0.92

L'utilizzo di convoluzioni 1D è più efficiente perché richiede meno parametri rispetto alle convoluzioni 2D, rendendole più adatte ad ambienti con risorse limitate.

È anche interessante prestare attenzione alla Matrice di Confusione 1D. Lo Il punteggio F1 per `yes` è 95% e per il `no`, 91%. Ciò era previsto da ciò che abbiamo visto con Feature Explorer (`no` e `unknown` a distanza ravvicinata). Nel tentativo di migliorare il risultato, si possono esaminare attentamente i risultati dei campioni con un errore.

Si ascoltino i campioni che sono andati male. Ad esempio, per `yes`, la maggior parte degli errori erano correlati a un sì pronunciato come “yeh”. Si possono acquisire campioni aggiuntivi e quindi riaddestrare il modello.

Andiamo sotto il cofano

Per capire cosa sta succedendo “sotto il cofano”, si può scaricare il set di dati pre-elaborato ((`MFCC training data`) dalla scheda `Dashboard` ed eseguire questo Jupyter Notebook, giocando con il codice o [Aprirlo in Colab]. Ad esempio, si può analizzare l’accuratezza per ogni epoca:

Test

Testando il modello con i dati riservati per il training (Test Data), abbiamo ottenuto un’accuratezza di circa il 76%.

Model testing results

ACCURACY
75.85%

	NO	NOISE	UNKNOWN	YES	UNCERTAIN
NO	57.8%	1.9%	27.8%	0.2%	12.2%
NOISE	0%	90.2%	2.3%	0.3%	7.2%
UNKNOWN	3.4%	3.7%	77.4%	0.7%	14.8%
YES	0.5%	5.0%	1.0%	82.3%	11.3%
F1 SCORE	0.72	0.89	0.70	0.90	

Ispezionando il punteggio F1, possiamo vedere che per YES abbiamo ottenuto 0.90, un risultato eccellente poiché prevediamo di utilizzare questa parola chiave come “trigger” primario per il nostro progetto KWS. Il risultato peggiore (0.70) è per UNKNOWN, il che è OK. Per NO, abbiamo ottenuto 0,72, come previsto, ma per migliorare questo risultato, possiamo spostare i campioni che non sono stati classificati correttamente nel set di dati di training e quindi ripetere il processo di training.

Classificazione Live

Possiamo procedere alla fase successiva del progetto, ma consideriamo anche che è possibile eseguire la `Classificazione live` utilizzando NiclaV o uno smartphone per catturare campioni dal vivo, testando il modello addestrato prima della distribuzione sul nostro dispositivo.

Distribuzione e Inferenza

L'EIS impacchetterà tutte le librerie necessarie, le funzioni di pre-elaborazione e i modelli addestrati, scaricandoli sul computer. Si va alla sezione `Deployment`, si seleziona `Arduino Library`, e, in basso, si sceglie `Quantized (Int8)` e si preme `Build`.

Quando si seleziona il pulsante `Build`, verrà creato un file zip che verrà scaricato sul computer. Sull'Arduino IDE, si va alla scheda `Sketch`, si seleziona l'opzione `Add .ZIP Library` e si sceglie il file .zip scaricato da EIS:

Ora è il momento di un vero test. Faremo delle inferenze completamente disconnesse da EIS. Usiamo l'esempio di codice NiclaV creato quando abbiamo distribuito la Libreria Arduino. Nell'IDE Arduino, si va alla scheda `File/Examples`, si cerca il progetto e si seleziona `nicla-vision/nicla-vision_microphone` (o `nicla-vision_microphone_continuous`)

Premere due volte il pulsante di reset per mettere NiclaV in modalità di avvio, caricare lo sketch sulla board e provare alcune inferenze reali:

Post-elaborazione

Ora che sappiamo che il modello funziona perché rileva le nostre parole chiave, modifichiamo il codice per vedere il risultato con NiclaV completamente offline (scollegato dal PC e alimentato da una batteria, un power bank o un alimentatore indipendente da 5 V). L'idea è che ogni volta che viene rilevata la parola chiave YES, si accende il LED verde; se si sente un NO, si accende il LED rosso, se è un UNKNOW, si accende il LED blu; e in presenza di rumore (nessuna parola chiave), i LED saranno SPENTI.
Dovremmo modificare uno degli esempi di codice. Facciamolo ora con `nicla-vision_microphone_continuous`.
Prima di tutto l'inizializzazione dei LED:

```
...
void setup()
{
        // Once you finish debugging your code, you can comment or delete the Serial part
of the code
    Serial.begin(115200);
    while (!Serial);
    Serial.println("Inferencing - Nicla Vision KWS with LEDs");

    // Pins for the built-in RGB LEDs on the Arduino NiclaV
    pinMode(LEDR, OUTPUT);
    pinMode(LEDG, OUTPUT);
    pinMode(LEDB, OUTPUT);

    // Ensure the LEDs are OFF by default.
    // Note: The RGB LEDs on the Arduino Nicla Vision
    // are ON when the pin is LOW, OFF when HIGH.
    digitalWrite(LEDR, HIGH);
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);
...
}
```

Creare due funzioni, la funzione `turn_off_leds()`, per spegnere tutti i LED RGB

```
/*
 * @brief      turn_off_leds function - turn-off all RGB LEDs
 */
void turn_off_leds(){
    digitalWrite(LEDR, HIGH);
    digitalWrite(LEDG, HIGH);
    digitalWrite(LEDB, HIGH);
}
```

Un'altra funzione `turn_on_led()` viene utilizzata per accendere i LED RGB in base al risultato più probabile del classificatore.

```
/*
 * @brief      turn_on_leds function used to turn on the RGB LEDs
 * @param[in]  pred_index
 *             no:       [0] ==> Red ON
 *             noise:    [1] ==> ALL OFF
 *             unknown:  [2] ==> Blue ON
 *             Yes:      [3] ==> Green ON
 */
void turn_on_leds(int pred_index) {
  switch (pred_index)
  {
    case 0:
      turn_off_leds();
      digitalWrite(LEDR, LOW);
      break;

    case 1:
      turn_off_leds();
      break;

    case 2:
      turn_off_leds();
      digitalWrite(LEDB, LOW);
      break;

    case 3:
      turn_off_leds();
      digitalWrite(LEDG, LOW);
      break;
  }
}
```

E modificare la parte `// print the predictions` del codice su `loop()`:

```
...

    if (++print_results >= (EI_CLASSIFIER_SLICES_PER_MODEL_WINDOW)) {
        // print the predictions
```

```
        ei_printf("Predictions ");
        ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
            result.timing.dsp, result.timing.classification, result.timing.anomaly);
        ei_printf(": \n");

        int pred_index = 0;     // Initialize pred_index
        float pred_value = 0;   // Initialize pred_value

        for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
            if (result.classification[ix].value > pred_value){
                pred_index = ix;
                pred_value = result.classification[ix].value;
            }

            // ei_printf("    %s: ", result.classification[ix].label);
            // ei_printf_float(result.classification[ix].value);
            // ei_printf("\n");
        }

        ei_printf("  PREDICTION: ==> %s with probability %.2f\n",
                  result.classification[pred_index].label, pred_value);
        turn_on_leds (pred_index);


#if EI_CLASSIFIER_HAS_ANOMALY == 1
        ei_printf("    anomaly score: ");
        ei_printf_float(result.anomaly);
        ei_printf("\n");
#endif

        print_results = 0;
    }
}
...
```

Il codice completo si trova tra i progetti GitHub.
Carica lo sketch sulla board e prova alcune inferenze reali. L'idea è che il LED verde sarà ACCESO ogni volta che viene rilevata la parola chiave YES, il rosso si accenderà per un NO e qualsiasi altra parola accenderà il LED blu. Tutti i LED dovrebbero essere spenti se è presente silenzio o rumore di fondo. Ricordare che la stessa procedura può "attivare" un dispositivo esterno per eseguire un'azione desiderata invece di accendere un LED, come abbiamo visto nell'introduzione.
https://youtu.be/25Rd76OTXLY

Conclusione

Si troveranno i notebook e i codici utilizzati in questo tutorial nel repository GitHub.

Prima di concludere, considerare che la classificazione dei suoni è più di una semplice voce. Ad esempio, si possono sviluppare progetti TinyML sul suono in diverse aree, come:

- **Sicurezza** (Rilevamento di vetri rotti, spari)
- **Industria** (Rilevamento di Anomalie)
- **Medicina** (Russamento, tosse, malattie polmonari)
- **Natura** (Controllo degli alveari, suono degli insetti, riduzione dei sacchetti di raccolta)

Risorse

- Sottoinsieme del Dataset dei Comandi Vocali di Google
- KWS MFCC Analysis Colab Notebook
- KWS_CNN_training Colab Notebook
- Codice di Post-elaborazione Arduino
- Progetto Edge Impulse

Classificazione del Movimento e Rilevamento delle Anomalie

DALL·E 3 Prompt: Illustrazione in stile fumetto anni '50 raffigurante una sala di ricerca sul movimento. Al centro della stanza c'è un container simulato utilizzato per il trasporto di merci su camion, barche e carrelli elevatori. Il container è dettagliato con rivetti e marcature tipiche dei cassoni di carico industriali. Attorno al container, la stanza è piena di attrezzature d'epoca, tra cui un oscilloscopio, vari array di sensori e grandi rotoli di carta di dati registrati. Le pareti sono adornate con poster educativi sulla sicurezza e la logistica dei trasporti. L'atmosfera generale della stanza è nostalgica e scientifica, con un pizzico di stile industriale.

Introduzione

I trasporti sono la spina dorsale del commercio globale. Milioni di container vengono trasportati ogni giorno tramite vari mezzi, come navi, camion e treni, verso destinazioni in tutto il mondo. Garantire il transito sicuro ed efficiente di questi container è un compito monumentale che richiede di sfruttare la tecnologia moderna e TinyML è senza dubbio una di queste.

In questo tutorial, lavoreremo per risolvere problemi reali relativi al trasporto. Svilupperemo un sistema di Motion Classification e Anomaly Detection utilizzando la scheda Arduino Nicla Vision, l'IDE Arduino e Edge Impulse Studio. Questo progetto ci aiuterà a comprendere come i container subiscono forze e movimenti diversi durante le varie fasi del trasporto, come il

transito terrestre e marittimo, il movimento verticale tramite carrelli elevatori e i periodi di stazionamento nei magazzini.

Obiettivi dell'Apprendimento

- Impostazione della scheda Arduino Nicla Vision
- Raccolta e pre-elaborazione dei dati
- Creazione del modello di classificazione del movimento
- Implementazione del rilevamento delle anomalie
- Test e Analisi nel Mondo Reale

Alla fine di questo tutorial, si avrà un prototipo funzionante in grado di classificare diversi tipi di movimento e rilevare anomalie durante il trasporto di container. Questa conoscenza può essere un trampolino di lancio per progetti più avanzati nel campo emergente di TinyML che coinvolge le vibrazioni.

Installazione e test dell'IMU

Per questo progetto, useremo un accelerometro. Come spiegato nel Tutorial, *Configurazione di Nicla Vision*, la Nicla Vision Board ha un **IMU a 6 assi** integrato: giroscopio 3D e accelerometro 3D, il LSM6DSOX. Verifichiamo se la libreria IMU LSM6DSOX è installata. In caso contrario, la si installa.

Poi si va su `Examples > Arduino_LSM6DSOX > SimpleAccelerometer` e si esegue il test dell'accelerometro. Si può verificare se funziona aprendo l'IDE Serial Monitor o Plotter. I valori sono in g (gravità terrestre), con un range di default di +/- 4g:

Definizione della Frequenza di Campionamento:

La scelta di una frequenza di campionamento appropriata è fondamentale per catturare le caratteristiche del movimento che interessa studiare. Il teorema di campionamento di Nyquist-Shannon afferma che la frequenza di campionamento dovrebbe essere almeno il doppio della componente di frequenza più alta nel segnale per ricostruirlo correttamente. Nel contesto della classificazione del movimento e del rilevamento delle anomalie per il trasporto, la scelta della frequenza di campionamento dipenderebbe da diversi fattori:

1. **Natura del movimento**: Diversi tipi di trasporto (terrestre, marittimo, ecc.) possono comportare diversi intervalli di frequenze di movimento. Movimenti più rapidi potrebbero richiedere frequenze di campionamento più elevate.
2. **Limitazioni hardware**: La scheda Arduino Nicla Vision e tutti i sensori associati potrebbero avere limitazioni sulla velocità con cui possono campionare i dati.
3. **Risorse di Calcolo**: Frequenze di campionamento più elevate genereranno più dati, il che potrebbe essere computazionalmente intensivo, particolarmente critico in un ambiente TinyML.
4. **Durata della Batteria**: Una frequenza di campionamento più elevata consumerà più energia. Se il sistema funziona a batteria, questa è una considerazione importante.
5. **Archiviazione Dati**: Un campionamento più frequente richiederà più spazio di archiviazione, un'altra considerazione cruciale per i sistemi embedded con memoria limitata.

In molte attività di riconoscimento delle attività umane, vengono comunemente utilizzate **frequenze di campionamento da 50 Hz a 100 Hz circa**. Dato che stiamo simulando scenari di trasporto, che in genere non sono eventi ad alta frequenza, una frequenza di campionamento in quell'intervallo (50-100 Hz) potrebbe essere un punto di partenza ragionevole.

Definiamo uno sketch che ci consentirà di acquisire i nostri dati con una frequenza di campionamento definita (ad esempio, 50 Hz):

```
/*
 * Based on Edge Impulse Data Forwarder Example (Arduino)
  - https://docs.edgeimpulse.com/docs/cli-data-forwarder
 * Developed by M.Rovai @11May23
 */

/* Include ---------------------------------------------------------------- */
```

```
#include <Arduino_LSM6DSOX.h>

/* Constant defines -------------------------------------------------------- */
#define CONVERT_G_TO_MS2 9.80665f
#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))

static unsigned long last_interval_ms = 0;
float x, y, z;

void setup() {
  Serial.begin(9600);
  while (!Serial);

  if (!IMU.begin()) {
    Serial.println("Failed to initialize IMU!");
    while (1);
  }
}

void loop() {
  if (millis() > last_interval_ms + INTERVAL_MS) {
    last_interval_ms = millis();

    if (IMU.accelerationAvailable()) {
      // Read raw acceleration measurements from the device
      IMU.readAcceleration(x, y, z);

      // converting to m/s2
      float ax_m_s2 = x * CONVERT_G_TO_MS2;
      float ay_m_s2 = y * CONVERT_G_TO_MS2;
      float az_m_s2 = z * CONVERT_G_TO_MS2;

      Serial.print(ax_m_s2);
      Serial.print("\t");
      Serial.print(ay_m_s2);
      Serial.print("\t");
      Serial.println(az_m_s2);
    }
  }
}
```

Caricando lo sketch e ispezionando il Serial Monitor, possiamo vedere che stiamo catturando 50 campioni al secondo.

Si noti che con la scheda Nicla appoggiata su un tavolo (con la telecamera rivolta verso il basso), l'asse z misura circa 9.8m/s^2, l'accelerazione terrestre prevista.

Il Caso di Studio: Trasporto Simulato di Container

Simuleremo il trasporto di container (o meglio di pacchi) attraverso diversi scenari per rendere questo tutorial più comprensibile e pratico. Utilizzando l'accelerometro integrato della scheda Arduino Nicla Vision, cattureremo i dati di movimento simulando manualmente le condizioni di:

1. Trasporto **Terrestre** (su strada o treno)
2. Trasporto **Marittimo**
3. Movimento Verticale tramite Carrello **Elevatore**
4. Periodo di **stazionamento** (inattivo) in un Magazzino

Dalle immagini sopra, possiamo definire per la nostra simulazione che i movimenti principalmente orizzontali (asse x o y) dovrebbero essere associati alla "classe Terrestre", i movimenti verticali (asse z) alla "classe di Sollevamento", nessuna attività alla "classe di Inattività" e il movimento su tutti e tre gli assi alla classe Marittima.

Raccolta Dati

Per la raccolta dati, possiamo avere diverse opzioni. In un caso reale, possiamo avere il nostro dispositivo, ad esempio, collegato direttamente a un contenitore e i dati raccolti su un file (ad esempio .CSV) e archiviati su una scheda SD (tramite connessione SPI) o un repository offline nel computer. I dati possono anche essere inviati in remoto a un repository nelle vicinanze, come un telefono cellulare, tramite Bluetooth (come fatto in questo progetto: Sensor DataLogger). Una volta che il dataset è stato raccolto e archiviato come file .CSV, può essere caricato su Studio utilizzando lo strumento CSV Wizard.

In questo video, si possono imparare modi alternativi per inviare dati a Edge Impulse Studio.

Collegamento del dispositivo a Edge Impulse

Collegheremo Nicla direttamente a Edge Impulse Studio, che verrà utilizzato anche per la pre-elaborazione dei dati, l'addestramento del modello, i test e la distribuzione. Per questo, ci sono due possibilità:

1. Scaricare il firmware più recente e collegarlo direttamente alla sezione `Raccolta Dati`.
2. Utilizzare il tool CLI Data Forwarder per acquisire i dati dal sensore e inviarli a Studio.

L'opzione 1 è più semplice, come abbiamo visto nell'esercitazione *Configurazione di Nicla Vision*, ma l'opzione 2 darà maggiore flessibilità per quanto riguarda l'acquisizione dei dati, come la definizione della frequenza di campionamento. Facciamolo con l'ultima.
Creare un nuovo progetto su Edge Impulse Studio (EIS) e collegarvi Nicla, seguendo questi passaggi:

1. Installare Edge Impulse CLI e Node.js sul computer.
2. Caricare uno sketch per l'acquisizione dati (quello discusso in precedenza in questo tutorial).
3. Utilizzare CLI Data Forwarder per acquisire dati dall'accelerometro di Nicla e inviarli a Studio, come mostrato in questo diagramma:

Avviare CLI Data Forwarder sul terminale, immettendo (se è la prima volta) il seguente comando:

```
$ edge-impulse-data-forwarder --clean
```

Quindi, immettere le proprie credenziali EI e scegliere il progetto, le variabili (ad esempio, *accX, accY* e *accZ*) e il nome del dispositivo (ad esempio, *NiclaV*):

```
marcelo_rovai — node ~/.npm-global/bin/edge-impulse-data-forwarder --clean — 88×16
Last login: Tue Oct 31 13:16:03 on ttys000
(base) marcelo_rovai@Marcelos-MacBook-Pro ~ % edge-impulse-data-forwarder --clean
Edge Impulse data forwarder v1.21.1
[? What is your user name or e-mail address (edgeimpulse.com)? rovai@mjrobot.org
[? What is your password? [hidden]
Endpoints:
    Websocket: wss://remote-mgmt.edgeimpulse.com
    API:       https://studio.edgeimpulse.com
    Ingestion: https://ingestion.edgeimpulse.com

[SER] Connecting to /dev/tty.usbmodem101
[SER] Serial is connected (00:2C:00:27:30:31:51:0C:39:31:35:32)
[WS ] Connecting to wss://remote-mgmt.edgeimpulse.com
[WS ] Connected to wss://remote-mgmt.edgeimpulse.com

? To which project do you want to connect this device? MJRoBot (Marcelo Rovai) / NICLA
Vision  Mov
ement Classification
[SER] Detecting data frequency...
[SER] Detected data frequency: 50Hz
[? 3 sensor axes detected (example values: [-1.26,-0.37,-9.79]). What do you want to cal
l them? Separate the names wit
h ',': accX, accY, accZ
? What name do you want to give this device? NiclaV
[WS ] Device "NiclaV" is now connected to project "NICLA Vision  Movement Classificatio
n". To connect to another project, run `edge-impulse-data-forwarder --clean`.
[WS ] Go to https://studio.edgeimpulse.com/studio/302078/acquisition/training to build
your machine learning model!
```

Andare alla sezione `Devices` sul progetto EI e verificare se il dispositivo è connesso (il punto dovrebbe essere verde):

Si può clonare il progetto sviluppato per questa esercitazione pratica: NICLA Vision Movement Classification.

Raccolta Dati

Nella sezione `Data Acquisition`, si dovrebbe vedere che la scheda `[NiclaV]` è connessa. Il sensore è disponibile: `[sensor with 3 axes (accX, accY, accZ)]` con una frequenza di campionamento di `[50Hz]`. Studio suggerisce una lunghezza di campionamento di `[10000]` ms (10s). L'ultima cosa rimasta è definire l'etichetta del campione. Cominciamo con `[terrestrial]`:

Terrestrial (pallet in un Camion o Treno), muovendosi orizzontalmente. Premere `[Start Sample]` e spostare il device orizzontalmente, mantenendo una direzione sopra il tavolo. Dopo 10s, i dati saranno caricati nello Studio. Ecco come è stato raccolto il campione:

Come previsto, il movimento è stato catturato principalmente nell'asse Y (verde). Nel blu, vediamo l'asse Z, circa -10 m/s^2 (Nicla ha la telecamera rivolta verso l'alto).
Come discusso in precedenza, dovremmo catturare dati da tutte e quattro le classi Transportation". Quindi, si immagini di avere un container con un accelerometro integrato nelle seguenti situazioni:

Maritime (pallet in barche in un oceano in tempesta). Il movimento viene catturato su tutti e tre gli assi:

Lift (Pallet movimentati verticalmente da un Carrello elevatore). Movimento catturato solo sull'asse Z:

Idle (Pallet in un magazzino). Nessun movimento rilevato dall'accelerometro:

Ad esempio, si possono catturare 2 minuti (dodici campioni di 10 secondi) per ciascuna delle quattro classi (per un totale di 8 minuti di dati). Utilizzando il menù `tre puntini` dopo ciascuno dei campioni, selezionarne 2, riservandoli per il set di Test. In alternativa, si può utilizzare lo strumento automatico `Train/Test Split tool` nella scheda `Danger Zone` della scheda `Dashboard`. Di seguito il set di dati risultante:

Una volta acquisito il dataset, lo si può esplorare più in dettaglio utilizzando Data Explorer, uno strumento visivo per trovare valori anomali o dati etichettati in modo errato (aiutando a correggerli). Data Explorer tenta prima di estrarre feature significative dai dati (applicando l'elaborazione del segnale e gli "embedding" della rete neurale) e poi utilizza un algoritmo di riduzione della dimensionalità come PCA o t-SNE per mappare queste feature in uno spazio 2D. Questo fornisce una panoramica immediata del dataset completo.

Nel nostro caso, il set di dati sembra OK (buona separazione). Ma il PCA mostra che possiamo avere problemi tra marittimo (verde) e sollevamento (arancione). Ciò è prevedibile, una volta su una barca, a volte il movimento può essere solo "verticale".

Impulse Design

Il passo successivo è la definizione del nostro Impulse, che prende i dati grezzi e usa l'elaborazione del segnale per estrarre le feature, passandole come tensore di input di un

learning block [blocco di apprendimento] per classificare nuovi dati. Si va in `Impulse Design` e `Create Impulse`. Studio suggerirà la progettazione di base. Aggiungiamo anche un secondo *Learning Block* per `Anomaly Detection`.

Questo secondo modello usa un modello K-means. Se immaginiamo di poter avere le nostre classi note come cluster, qualsiasi campione che si adatterebbe potrebbe essere un "outlier", un'anomalia come un container che rotola fuori da una nave in mare o cade da un carrello elevatore.

La frequenza di campionamento dovrebbe essere catturata automaticamente, in caso contrario, inserirla: `[50]` Hz. Studio suggerisce una *Window Size* di 2 secondi (`[2000]` ms) con una *sliding window* di `[20]` ms. Ciò che stiamo definendo in questa fase è che pre-

elaboreremo i dati catturati (dati di serie temporali), creando un dataset tabellare (caratteristiche) che saranno l'input per un classificatore di reti neurali (DNN) e un modello di rilevamento delle anomalie (K-Means), come mostrato di seguito:

Analizziamo attentamente quei passaggi e i parametri per capire meglio cosa stiamo facendo qui.

Panoramica sulla Pre-Elaborazione dei Dati

La pre-elaborazione dei dati consiste nell'estrapolare le feature dal set di dati acquisito con l'accelerometro, il che implica l'elaborazione e l'analisi dei dati grezzi. Gli accelerometri misurano l'accelerazione di un oggetto lungo uno o più assi (in genere tre, indicati come X, Y e Z). Queste misure possono essere utilizzate per comprendere vari aspetti del movimento dell'oggetto, come pattern di movimento e vibrazioni.

I dati grezzi dell'accelerometro possono essere rumorosi e contenere errori o informazioni irrilevanti. Le fasi di pre-elaborazione, come il filtraggio e la normalizzazione, possono pulire e standardizzare i dati, rendendoli più adatti all'estrazione di feature. Nel nostro caso, dovremmo dividere i dati in segmenti più piccoli o **windows** [finestre]. Ciò può aiutare a concentrarsi su eventi o attività specifici all'interno del dataset, rendendo l'estrazione di feature più gestibile e significativa. La scelta della **window size** e della sovrapposizione (**window increase**) dipende dall'applicazione e dalla frequenza degli eventi di interesse. Come regola generale, dovremmo provare a catturare un paio di "cicli di dati".

> Con un "sampling rate" (SR) [frequenza di campionamento] di 50 Hz e una dimensione della finestra di 2 secondi, otterremo 100 campioni per asse, o 300 in totale (3 assi x 2 secondi x 50 campioni). Faremo scorrere questa finestra ogni 200 ms, creando un set di dati più grande in cui ogni istanza ha 300 feature grezze.

Una volta che i dati sono stati pre-elaborati e segmentati, si possono estrarre feature che descrivono le caratteristiche del movimento. Alcune feature tipiche estratte dai dati dell'accelerometro includono:

- Le feature del **Time-domain** descrivono le proprietà statistiche dei dati all'interno di ciascun segmento, come media, mediana, deviazione standard, asimmetria, curtosi e tasso di attraversamento dello zero.
- Le feature **Frequency-domain** si ottengono trasformando i dati nel dominio della frequenza utilizzando tecniche come la Fast Fourier Transform (FFT). Alcune feature tipiche del dominio della frequenza includono lo spettro di potenza, l'energia spettrale, le frequenze dominanti (ampiezza e frequenza) e l'entropia spettrale.
- Le feature del dominio **Time-frequency** combinano le informazioni del dominio del tempo e della frequenza, come la Short-Time Fourier Transform (STFT) o la Discrete Wavelet Transform (DWT). Possono fornire una comprensione più dettagliata di come il contenuto di frequenza del segnale cambia nel tempo.

In molti casi, il numero di feature estratte può essere elevato, il che può portare a un overfitting o a una maggiore complessità computazionale. Le tecniche di selezione delle feature, come le informazioni reciproche, i metodi basati sulla correlazione o l'analisi delle componenti principali (PCA), possono aiutare a identificare le feature più rilevanti per una determinata applicazione e ridurre la dimensionalità del dataset. Studio può aiutare con tali calcoli di importanza delle feature.

Feature Spettrali di EI Studio

La pre-elaborazione dei dati è un'area impegnativa per il machine learning embedded; tuttavia, Edge Impulse aiuta a superarla con la sua fase di pre-elaborazione del segnale digitale (DSP) e, più specificamente, con lo Spectral Features Block.
In Studio, il dataset grezzo raccolto sarà l'input di un blocco di Spectral Analysis, che è eccellente per analizzare il movimento ripetitivo, come i dati degli accelerometri. Questo blocco eseguirà un DSP (Digital Signal Processing), estraendo feature come la FFT o le Wavelet.
Per il nostro progetto, una volta che il segnale temporale è continuo, dovremmo usare FFT con, ad esempio, una lunghezza di `[32]`.

Le feature **Time Domain Statistical** per asse/canale sono:

- RMS: 1 feature
- Skewness [Asimmetria]: 1 feature
- Curtosi: 1 feature

Le feature **Frequency Domain Spectral features** per asse/canale sono:

- Potenza Spettrale: 16 feature (lunghezza FFT/2)
- Asimmetria: 1 feature
- Curtosi: 1 feature

Quindi, per una lunghezza FFT di 32 punti, l'output risultante dello "Spectral Analysis Block" sarà di 21 feature per asse (un totale di 63 feature).

> Si può scoprire di più su come viene calcolata ogni feature scaricando il notebook Edge Impulse - Spectral Features Block Analysis TinyML under the hood: Spectral Analysis o aprendolo direttamente su Google CoLab.

Generazione di feature

Una volta capito cosa fa la pre-elaborazione, è il momento di finire il lavoro. Quindi, prendiamo i dati grezzi (tipo serie temporale) e convertiamoli in dati tabellari. Per farlo, si va

alla sezione `Spectral Features` nella scheda `Parameters`, si definiscono i parametri principali come discusso nella sezione precedente (`[FFT]` con `[32]` punti), e si seleziona `[Save Parameters]`:

Nel menù in alto, si seleziona l'opzione `Generate Features` e il pulsante `Generate Features`. Ogni dato della finestra di 2 secondi verrà convertito in un punto dati di 63 feature.

> Feature Explorer mostrerà tali dati in 2D utilizzando UMAP. Uniform Manifold Approximation and Projection (UMAP) è una tecnica di riduzione delle dimensioni che può essere utilizzata per la visualizzazione in modo simile a t-SNE, ma è applicabile anche per la riduzione generale delle dimensioni non lineari.

La visualizzazione consente di verificare che dopo la generazione delle feature, le classi presenti mantengano la loro eccellente separazione, il che indica che il classificatore dovrebbe funzionare bene. Facoltativamente, si può analizzare quanto è importante ciascuna delle feature per una classe rispetto alle altre.

Addestramento dei Modelli

Il nostro classificatore sarà una Dense Neural Network (DNN) che avrà 63 neuroni sul suo layer di input, due layer nascosti con 20 e 10 neuroni e un layer di output con quattro neuroni (uno per ogni classe), come mostrato qui:

Come iperparametri, useremo un Learning Rate di `[0.005]`, una dimensione Batch di `[32]` e `[20]`% di dati per la convalida per `[30]` epoche. Dopo l'addestramento, possiamo vedere che l'accuratezza è del 98.5%. Il costo della memoria e della latenza è esiguo.

Per la "Anomaly Detection", sceglieremo le feature suggerite che sono esattamente le più importanti nell'estrazione delle feature, più l'RMS accZ. Il numero di cluster sarà `[32]`, come suggerito da Studio:

Test

Possiamo verificare come si comporterà il nostro modello con dati sconosciuti utilizzando il 20% dei dati lasciati da parte durante la fase di acquisizione dei dati. Il risultato è stato quasi del 95%, il che è positivo. Si può sempre lavorare per migliorare i risultati, ad esempio, per capire cosa è andato storto con uno dei risultati sbagliati. Se si tratta di una situazione unica, la si può aggiungere al set di dati di training e quindi ripeterla.
La soglia minima di default per un risultato considerato incerto è `[0.6]` per la classificazione e `[0.3]` per l'anomalia. Una volta che abbiamo quattro classi (la loro somma di output dovrebbe essere 1.0), si può anche impostare una soglia inferiore per una classe da considerare valida (ad esempio, 0.4). Si possono `Impostare le soglie di confidenza` nel menù `tre puntini`, oltre al pulsante `Classify all`.

Puoi anche eseguire "Live Classification" col dispositivo (che dovrebbe essere ancora connesso a Studio).

> Da tenere presente che qui, si cattureranno dati reali col dispositivo e si caricheranno in Studio, dove verrà presa un'inferenza utilizzando il modello addestrato (ma il **modello NON è nel device**).

Distribuzione

È il momento di distribuire il blocco di pre-elaborazione e il modello addestrato al Nicla. Studio impacchetterà tutte le librerie necessarie, le funzioni di pre-elaborazione e i modelli addestrati, scaricandoli sul computer. Si deve selezionare l'opzione `Arduino Library`, e in basso si può scegliere `Quantized (Int8)` o `Unoptimized (float32)` e `[Build]`. Verrà creato un file Zip e scaricato sul computer.

Su Arduino IDE, si va alla scheda `Sketch`, si seleziona `Add.ZIP Library` e si sceglie il file .zip scaricato da Studio. Un messaggio apparirà nel terminale IDE: `Library installed`.

Inferenza

Ora è il momento di un vero test. Faremo inferenze completamente scollegate da Studio. Modifichiamo uno degli esempi di codice creati quando si distribuisce la libreria Arduino. Nell'IDE Arduino, si va alla scheda `File/Examples` e si cerca il progetto, e negli esempi, si seleziona `Nicla_vision_fusion`:

Notare che il codice creato da Edge Impulse considera un approccio *sensor fusion* in cui vengono utilizzati IMU (accelerometro e giroscopio) e ToF. All'inizio del codice, ci sono le librerie relative al nostro progetto, IMU e ToF:

```
/* Includes ------------------------------------------------------------------ */
#include <NICLA_Vision_Movement_Classification_inferencing.h>
#include <Arduino_LSM6DSOX.h> //IMU
#include "VL53L1X.h" // ToF
```

> Si può mantenere il codice in questo modo per i test perché il modello addestrato utilizzerà solo le funzionalità pre-elaborate dall'accelerometro. Ma si consideri che si scriverà il codice solo con le librerie necessarie per un progetto reale.

E questo è tutto!
Ora si può caricare il codice sul dispositivo e procedere con le inferenze. Si preme due volte il pulsante `[RESET]` di Nicla per metterlo in modalità boot (disconnetterlo da Studio se è ancora connesso) e caricare lo sketch sulla board.
Ora si devono provare diversi movimenti con la scheda (simili a quelli eseguiti durante l'acquisizione dei dati), osservando il risultato dell'inferenza di ciascuna classe sul Serial Monitor:

- **Classi Idle e lift:**

- **Maritime e terrestrial:**

Notare che in tutte le situazioni sopra, il valore di `anomaly score` era inferiore a 0.0. Provare un nuovo movimento che non faceva parte del dataset originale, ad esempio, “facendo rotolare” la Nicla, rivolta verso la telecamera capovolta, come un contenitore che cade da una barca o persino un incidente in barca:

- **Rilevamento delle Anomalie**

In questo caso, l’anomalia è molto più grande, oltre 1.00

Post-elaborazione

Ora che sappiamo che il modello funziona poiché rileva i movimenti, suggeriamo di modificare il codice per vedere il risultato con il NiclaV completamente offline (scollegato dal PC e alimentato da una batteria, un power bank o un alimentatore indipendente da 5V). L'idea è di fare lo stesso del progetto KWS: se viene rilevato un movimento specifico, potrebbe accendersi un LED specifico. Ad esempio, se viene rilevato un movimento *terrestre*, si accenderà il LED verde; se è un movimento *marittimo*, si accenderà il LED rosso, se è un *ascensore*, si accenderà il LED blu; e se non viene rilevato alcun movimento *(inattivo*), i LED saranno SPENTI. Si può anche aggiungere una condizione quando viene rilevata un'anomalia, in questo caso, ad esempio, può essere utilizzato un colore bianco (tutti i LED e si accendono contemporaneamente).

Conclusione

> I notebook e i codici utilizzati in questo tutorial pratico si trovano nel repository GitHub.

Prima di concludere, considerare che la classificazione del movimento e il rilevamento degli oggetti possono essere utilizzati in molte applicazioni in vari domini. Ecco alcune delle potenziali applicazioni:

Applicazioni di Casi

Industriale e Manifatturiero

- **Manutenzione Predittiva**: Rilevamento di anomalie nel movimento dei macchinari per prevedere guasti prima che si verifichino.
- **Controllo Qualità**: Monitoraggio del movimento di linee di assemblaggio o bracci robotici per la valutazione della precisione e il rilevamento delle deviazioni dal pattern di movimenti standard.
- **Logistica di Magazzino**: Gestione e tracciamento del movimento delle merci con sistemi automatizzati che classificano diversi tipi di movimento e rilevano anomalie nella movimentazione.

Assistenza Sanitaria

- **Monitoraggio Pazienti**: Rilevamento di cadute o movimenti anomali negli anziani o in coloro che hanno problemi di mobilità.
- **Riabilitazione**: Monitoraggio dei progressi dei pazienti in fase di recupero da infortuni tramite classificazione dei pattern di movimento durante le sedute di fisioterapia.
- **Riconoscimento Attività**: Classificazione dei tipi di attività fisica per applicazioni di fitness o monitoraggio dei pazienti.

Elettronica di Consumo

- **Gesture Control**: Interpretazione di movimenti specifici per controllare i dispositivi, come accendere le luci con un gesto della mano.
- **Gaming**: Miglioramento delle esperienze di gioco con input controllati dal movimento.

Trasporti e Logistica

- **Telematica dei Veicoli**: Monitoraggio del movimento del veicolo per comportamenti insoliti come frenate brusche, curve strette o incidenti.
- **Monitoraggio del Carico**: Garantire l'integrità delle merci durante il trasporto rilevando movimenti insoliti che potrebbero indicare manomissioni o cattiva gestione.

Città Intelligenti e Infrastrutture

- **Monitoraggio Strutturale**: Rilevare vibrazioni o movimenti all'interno delle strutture che potrebbero indicare potenziali guasti o necessità di manutenzione.
- **Gestione del Traffico**: Analizzare il flusso di pedoni o veicoli per migliorare la mobilità e la sicurezza urbana.

Sicurezza e Sorveglianza

- **Rilevamento Intrusi**: Rilevare pattern di movimento tipici di accessi non autorizzati o altre violazioni della sicurezza.
- **Monitoraggio della Fauna Selvatica**: Rilevare bracconieri o movimenti anomali di animali in aree protette.

Agricoltura

- **Monitoraggio delle Attrezzature**: Monitoraggio delle prestazioni e dell'utilizzo di macchinari agricoli.
- **Analisi del Comportamento Animale**: Monitoraggio dei movimenti del bestiame per rilevare comportamenti che indicano problemi di salute o stress.

Monitoraggio Ambientale

- **Attività Sismica**: Rilevamento di pattern di movimento irregolari che precedono terremoti o altri eventi geologicamente rilevanti.
- **Oceanografia**: Studio di pattern di onde o movimenti marini per scopi di ricerca e sicurezza.

Custodia per Nicla 3D

Per applicazioni reali, come alcuni hanno descritto in precedenza, possiamo aggiungere una custodia al nostro dispositivo ed Eoin Jordan, di Edge Impulse, ha sviluppato un'ottima custodia indossabile e per la salute delle macchine per la gamma di schede Nicla. Funziona con un magnete da 10 mm, viti da 2M e una cinghia da 16mm per scenari di utilizzo per la salute umana e delle macchine. Ecco il link: Arduino Nicla Voice e Vision Wearable Case.

Le applicazioni per la classificazione del movimento e il rilevamento delle anomalie sono estese e Arduino Nicla Vision è adatto per scenari in cui il basso consumo energetico e l'edge processing sono vantaggiosi. Il suo piccolo fattore di forma e l'efficienza nell'elaborazione lo rendono una scelta ideale per l'implementazione di applicazioni portatili e remote in cui l'elaborazione in tempo reale è fondamentale e la connettività potrebbe essere limitata.

Risorse

- Codice Arduino
- Edge Impulse Spectral Features Block Colab Notebook
- Progetto Edge Impulse

XIAO ESP32S3

Questi laboratori offrono un'opportunità unica per acquisire esperienza pratica con i sistemi di apprendimento automatico (ML). A differenza del lavoro con modelli di grandi dimensioni che richiedono risorse su scala di data center, questi esercizi consentono di interagire direttamente con hardware e software utilizzando TinyML. Questo approccio pratico offre una comprensione tangibile delle sfide e delle opportunità nell'implementazione dell'IA, sebbene su piccola scala. Tuttavia, i principi sono in gran parte gli stessi che si incontrerebbero quando si lavora con sistemi più grandi.

XIAO ESP32S3 Sense. Fonte: SEEED Studio

21.4 Prerequisiti

- **XIAO ESP32S3 Sense Board**: Si deve avere la scheda XIAO ESP32S3 Sense.
- **USB-C Cable**: Per collegare la scheda al computer.
- **Rete**: Con accesso a Internet per scaricare il software necessario.
- **SD Card e SD card Adapter**: Per salvare audio e immagini (opzionale).

21.5 Setup

- Setup della XIAO ESP32S3

21.6 Esercizi

Modalità	Attività	Descrizione	Link
Visione	Classificazione delle immagini	Impara a classificare le immagini	Link
Visione	Rilevamento di Oggetti	Implementa il rilevamento oggetti	Link
Suono	Individuazione Parole Chiave	Esplora sistemi di riconoscimento vocale	Link
IMU	Classificazione del Movimento e Rilevamento Anomalie	Classifica i dati di movimento e rileva anomalie	Link

Setup

DALL·E prompt - Disegno in stile cartone animato anni '50 di una scheda XIAO ESP32S3 con un modulo telecamera distintivo, come mostrato nell'immagine fornita. La scheda è posizionata su un classico tavolo da laboratorio con vari sensori, tra cui un microfono. Dietro la scheda, uno schermo di computer vintage visualizza l'IDE Arduino in colori tenui, con codice incentrato sulle configurazioni dei pin LED e inferenza di apprendimento automatico per i comandi vocali. Il Serial Monitor sull'IDE mostra output che rilevano comandi vocali come 'yes' e 'no'. La scena fonde il fascino retrò dei laboratori di metà secolo con l'elettronica moderna.

Introduzione

XIAO ESP32S3 Sense è la scheda di sviluppo conveniente di Seeed Studio, che integra un sensore della fotocamera, un microfono digitale e il supporto per schede SD. Combinando la potenza di elaborazione ML embedded e la capacità fotografica, questa scheda di sviluppo è un ottimo strumento per iniziare con TinyML (IA vocale e visiva).

Caratteristiche Principali di XIAO ESP32S3 Sense

- **Potente Scheda MCU**: Incorpora il chip del processore Xtensa dual-core ESP32S3 a 32 bit che funziona fino a 240 MHz, più porte di sviluppo montate, supporto per Arduino/MicroPython
- **Funzionalità Avanzate**: Sensore della fotocamera OV2640 staccabile per una risoluzione di 1600 * 1200, compatibile con il sensore della fotocamera OV5640, integra un microfono digitale aggiuntivo
- **Progetto di Alimentazione Elaborato**: La capacità di gestione della carica della batteria al litio offre quattro modelli di consumo energetico, che consentono la modalità di "deep sleep" [sospensione profonda] con un consumo energetico basso fino a 14μA
- **Grande Memoria per più Possibilità**: Offre 8 MB di PSRAM e 8 MB di FLASH, supportando uno slot per schede SD di memoria FAT esterno da 32 GB
- **Prestazioni RF Eccezionali**: Supporta la comunicazione wireless duale Wi-Fi e BLE a 2,4 GHz, supporta la comunicazione remota a 100m+ se connesso all'antenna U.FL
- **Design Compatto delle Dimensioni di un Pollice**: 21 x 17,5 mm, adotta il fattore di forma classico di XIAO, adatto per progetti con spazio limitato come i dispositivi indossabili

Di seguito è riportato il pinout generale della scheda:

Per maggiori dettagli, fare riferimento alla pagina Seeed Studio WiKi:
https://wiki.seeedstudio.com/xiao_esp32s3_getting_started/

Installazione di XIAO ESP32S3 Sense su Arduino IDE

Su Arduino IDE, si va su **File > Preferences** e si inserisce l'URL:
https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_dev_index.json

nel campo ==> **Additional Boards Manager URLs**

Poi si apre il gestore delle schede. Si va su **Tools** > **Board** > **Boards Manager...** e immettendo *esp32*. Selezionare e installare il pacchetto più aggiornato e stabile (evitare le versioni *alpha*) :

> ⚠ **Attenzione**
>
> Le versioni Alpha (ad esempio, 3.x-alpha) non funzionano correttamente con XIAO ed Edge Impulse. Utilizzare invece l'ultima versione stabile (ad esempio, 2.0.11).

Su **Tools**, selezionare la Board (**XIAO ESP32S3**):

Ultimo ma non meno importante, scegliere la **Porta** a cui è collegato l'ESP32S3.
Ecco fatto! Il dispositivo dovrebbe funzionare. Facciamo qualche test.

Test della scheda con BLINK

XIAO ESP32S3 Sense ha un LED integrato che è collegato a GPIO21. Quindi, si può eseguire lo sketch Blink così com'è (utilizzando la costante Arduino `LED_BUILTIN`) o modificando di conseguenza lo sketch Blink:

```
#define LED_BUILT_IN 21

void setup() {
  pinMode(LED_BUILT_IN, OUTPUT); // Set the pin as output
}

// Remember that the pin work with inverted logic
// LOW to Turn on and HIGH to turn off
void loop() {
  digitalWrite(LED_BUILT_IN, LOW); //Turn on
  delay (1000); //Wait 1 sec
  digitalWrite(LED_BUILT_IN, HIGH); //Turn off
  delay (1000); //Wait 1 sec
}
```

> Notare che i pin funzionano con logica invertita: BASSO per accendere e ALTO per spegnere.

Collegamento del Modulo Sense (Scheda di Espansione)

Quando viene acquistata, la scheda di espansione è separata da quella principale, ma installarla è molto semplice. Si deve allineare il connettore sulla scheda di espansione col connettore B2B sullo XIAO ESP32S3, premerlo con forza e quando si sente un “clic”, l’installazione è completa.
Come commentato nell’introduzione, la scheda di espansione, o la parte “sense” del dispositivo, ha una fotocamera OV2640 da 1600x1200, uno slot per schede SD e un microfono digitale.

Test del Microfono

Cominciamo con il rilevamento del suono. Si va nei progetti GitHub e si scarica lo sketch: XIAOEsp2s3_Mic_Test poi lo si esegue sull’IDE Arduino:

Quando si produce un suono, è possibile verificarlo sul Serial Plotter.

Salvare il suono registrato (file audio .wav) su una scheda microSD.
Ora, il lettore di schede SD integrato può salvare file audio .wav. Per farlo, dobbiamo abilitare la PSRAM XIAO.

> ESP32-S3 ha solo poche centinaia di kilobyte di RAM interna sul chip MCU. Questo può essere insufficiente per alcuni scopi, quindi fino a 16 MB di PSRAM esterna (RAM pseudo-statica) possono essere collegati al chip flash SPI. La memoria esterna è incorporata nella mappa di memoria e, con alcune restrizioni, è utilizzabile allo stesso modo della RAM dati interna.

Per iniziare, inserire la scheda SD su XIAO come mostrato nella foto qui sotto (la scheda SD deve essere formattata in **FAT32**).

- Scaricare lo sketch Wav_Record, che si può trovare su GitHub.
- Per eseguire il codice (Wav Record), è necessario utilizzare la funzione PSRAM del chip ESP-32, quindi la si deve attivare prima di caricare: Tools>PSRAM: “OPI PSRAM”>OPI PSRAM

- Eseguire il codice `Wav_Record.ino`
- Questo programma viene eseguito solo una volta dopo che l'utente accende il monitor seriale. Registra per 20 secondi e salva il file su una scheda microSD come "arduino_rec.wav."
- Quando "." viene emesso ogni 1 secondo nel monitor seriale, l'esecuzione del programma è terminata e si può riprodurre il file audio registrato con l'aiuto di un lettore di schede.

La qualità del suono è eccellente!

La spiegazione di come funziona il codice va oltre lo scopo di questo tutorial, ma c'è un'eccellente descrizione sulla pagina wiki.

Test della Fotocamera

Per testare la fotocamera, si deve scaricare la cartella take_photos_command da GitHub. La cartella contiene lo sketch (.ino) e due file .h con i dettagli della fotocamera.

- Eseguire il codice: `take_photos_command.ino`. Aprire il Serial Monitor e inviare il comando `capture` per catturare e salvare l'immagine sulla scheda SD:

Verificare che `[Both NL & CR]` sia selezionato sul Serial Monitor.

Ecco un esempio di una foto scattata:

Test del WiFi

Uno dei punti di forza di XIAO ESP32S3 è la sua capacità WiFi. Quindi, testiamo la sua radio eseguendo la scansione delle reti Wi-Fi attorno ad essa. Lo si può fare eseguendo uno degli esempi di codice sulla scheda.

Si va su Arduino IDE Examples e si cerca **WiFI ==> WiFIScan**
Si dovrebbero vedere le reti Wi-Fi (SSID e RSSI) nel raggio d'azione del dispositivo sul monitor seriale. Ecco cosa abbiamo ottenuto in laboratorio:

Un Semplice Server WiFi (accensione/spegnimento del LED)
Testiamo la capacità del dispositivo di comportarsi come un server WiFi. Ospiteremo una semplice pagina sul dispositivo che invia comandi per accendere e spegnere il LED integrato di XIAO.
Come prima, si va su GitHub per scaricare la cartella usando lo sketch SimpleWiFiServer.
Prima di eseguire lo sketch, si devono inserire le credenziali di rete:

```
const char* ssid     = "Your credentials here";
const char* password = "Your credentials here";
```

Si può monitorare il funzionamento del server con Serial Monitor.

Prendere l'indirizzo IP e inserirlo nel browser:

Si vedrà una pagina con link che possono accendere e spegnere il LED integrato del XIAO.

Streaming video sul Web

Ora che sappiamo che si possono inviare comandi dalla pagina Web al dispositivo, facciamo il contrario. Prendiamo l'immagine catturata dalla telecamera e la trasmettiamo in streaming su una pagina Web:

Scaricare da GitHub la cartella che contiene il codice: XIAO-ESP32S3-Streeming_Video.ino.

> Ricordarsi che la cartella contiene il file .ino e un paio di file .h necessari per gestire la telecamera.

Inserire le credenziali ed eseguire lo sketch. Sul monitor seriale, si può trovare l'indirizzo della pagina da inserire nel browser:

Apri la pagina sul browser (attendere qualche secondo per avviare lo streaming). Tutto qui.

Mettere in streaming ciò che la telecamera “vede” può essere importante quando la si posiziona per catturare un set di dati per un progetto ML (ad esempio, usando il codice “take_phots_commands.ino”.
Naturalmente, possiamo fare entrambe le cose contemporaneamente: mostrare ciò che la telecamera vede sulla pagina e inviare un comando per catturare e salvare l’immagine sulla scheda SD. Per questo, si può usare il codice Camera_HTTP_Server_STA, scaricabile da GitHub.

Il programma eseguirà le seguenti attività:

- Imposta la fotocamera in modalità di output JPEG.
- Crea una pagina Web (ad esempio ==> http://192.168.4.119//). L'indirizzo corretto verrà visualizzato sul Serial Monitor.
- Se server.on ("/capture", HTTP_GET, serverCapture), il programma scatta una foto e la invia al Web.
- È possibile ruotare l'immagine sulla pagina Web utilizzando il pulsante [ROTATE]
- Il comando [CAPTURE] visualizzerà solo l'anteprima dell'immagine sulla pagina Web, mostrandone le dimensioni sul Serial Monitor
- Il comando `[SAVE]` salverà un'immagine sulla scheda SD e la mostrerà sul browser.
- Le immagini salvate seguiranno una denominazione sequenziale (image1.jpg, image2.jpg.

Questo programma può catturare un set di dati di immagini con un progetto di classificazione delle immagini.

Esaminate il codice; sarà più facile capire come funziona la fotocamera. Questo codice è stato sviluppato sulla base del fantastico tutorial di Rui Santos ESP32-CAM Take Photo and Display in Web Server, che invito tutti a visitare.

Uso di CameraWebServer

Nell'IDE Arduino, si va su `File > Examples > ESP32 > Camera` e si seleziona `CameraWebServer`
Si devono anche commentare tutti i modelli di fotocamere, eccetto i pin del modello XIAO:
`#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM`
Non dimenticare `Tools` per abilitare la PSRAM.
Inserire le credenziali wifi e caricare il codice sul dispositivo:

Se il codice viene eseguito correttamente, si vedrà l'indirizzo sul monitor seriale:

```
WiFi connected
[  1946][I][app_httpd.cpp:1361] startCameraServer(): Starting web server on port: '80'
[  1948][I][app_httpd.cpp:1379] startCameraServer(): Starting stream server on port: '81'
Camera Ready! Use 'http://192.168.86.250' to connect
```

Copiare l'indirizzo sul browser e attendere che la pagina venga caricata. Selezionare la risoluzione della telecamera (ad esempio, QVGA) e selezionare `[START STREAM]`. Attendi qualche secondo/minuto, a seconda della connessione. Utilizzando il pulsante `[Save]`, si può salvare un'immagine nell'area download del computer.

Ecco fatto! Si possono salvare le immagini direttamente sul computer per usarle nei progetti.

Conclusione

The XIAO ESP32S3 Sense è flessibile, economico e facile da programmare. Con 8 MB di RAM, la memoria non è un problema e il dispositivo può gestire molte attività di post-elaborazione, inclusa la comunicazione.
L'ultima versione del codice si trova nel repository GitHub: XIAO-ESP32S3-Sense.

Risorse

- XIAO ESP32S3 Code

Classificazione delle Immagini

Immagine di Marcelo Rovai

Introduzione

Sempre di più, ci troviamo di fronte a una rivoluzione dell'intelligenza artificiale (IA) in cui, come affermato da Gartner, **Edge AI** ha un potenziale di impatto molto elevato, ed **è ora**!

In prima linea nel Radar delle Tecnologie Emergenti c'è il linguaggio universale di Edge Computer Vision. Quando esaminiamo il Machine Learning (ML) applicato alla visione, il primo concetto che ci accoglie è la classificazione delle immagini, una specie "Hello World" del ML che è sia semplice che profondo!
Seeed Studio XIAO ESP32S3 Sense è un potente strumento che combina il supporto per fotocamera e scheda SD. Con la sua potenza di elaborazione ML embedded e la capacità di fotografia, è un ottimo punto di partenza per esplorare l'intelligenza artificiale per la visione TinyML.

Un progetto TinyML di Motion Classification: Frutta contro Verdura

L'idea di base del nostro progetto sarà quella di addestrare un modello e procedere con l'inferenza su XIAO ESP32S3 Sense. Per l'addestramento, dovremmo trovare alcuni dati **(in effetti, tonnellate di dati!**).
Ma prima di tutto, abbiamo bisogno di un obiettivo! Cosa vogliamo classificare?

Con TinyML, un set di tecniche associate all'inferenza di apprendimento automatico su dispositivi embedded, dovremmo limitare la classificazione a tre o quattro categorie a causa di limitazioni (principalmente della memoria). Differenziamo **mele** da **banane** e **patate** (si possono provare altre categorie)**.**
Quindi, cerchiamo un set di dati specifico che includa immagini da quelle categorie. Kaggle è un buon inizio:
https://www.kaggle.com/kritikseth/fruit-and-vegetable-image-recognition
Questo set di dati contiene immagini dei seguenti alimenti:

- **Frutta** - *banana, mela*, pera, uva, arancia, kiwi, anguria, melograno, ananas, mango.
- **Verdura** - cetriolo, carota, peperone, cipolla, *patata,* limone, pomodoro, ravanello, barbabietola, cavolo, lattuga, spinaci, soia, cavolfiore, peperone, peperoncino, rapa, mais, mais dolce, patata dolce, paprika, jalepeño, zenzero, aglio, piselli, melanzane.

Ogni categoria è suddivisa in **train** [addestramento] (100 immagini), **test** (10 immagini) e **validation** (10 immagini).

- Scaricare il dataset dal sito Web di Kaggle e metterlo sul computer.

  Facoltativamente, si possono aggiungere alcune foto fresche di banane, mele e patate direttamente dalla cucina di casa, utilizzando, ad esempio, il codice discusso nel lab della configurazione.

Addestramento del modello con Edge Impulse Studio

Utilizzeremo Edge Impulse Studio per addestrare il modello. Come si saprà, Edge Impulse è una piattaforma di sviluppo leader per l'apprendimento automatico su dispositivi edge. Inserire le credenziali del proprio account (o crearne uno gratuito) su Edge Impulse. Quindi, creare un nuovo progetto:

Raccolta Dati

Successivamente, nella sezione `UPLOAD DATA`, caricare dal computer i file delle categorie scelte:

Sarebbe meglio se ora si avesse il proprio set di dati di training suddiviso in tre classi di dati:

Si possono caricare dati extra per ulteriori test del modello o dividere i dati di training. Lo lasceremo così com'è per utilizzare più dati possibili.

Impulse Design

Un impulso prende dati grezzi (in questo caso, immagini), estrae feature (ridimensiona le immagini) e poi utilizza un blocco di apprendimento per classificare nuovi dati.

La classificazione delle immagini è l'uso più comune del deep learning, ma per portare a termine questo compito dovrebbero essere utilizzati molti dati. Abbiamo circa 90 immagini per ogni categoria. Questo numero è sufficiente? No, per niente! Avremo bisogno di migliaia di immagini per "insegnare o modellare" per distinguere una mela da una banana. Ma possiamo risolvere questo problema riaddestrando un modello precedentemente addestrato con migliaia di immagini. Chiamiamo questa tecnica "Transfer Learning" (TL).

Col TL, possiamo mettere a punto un modello di classificazione delle immagini pre-addestrato sui nostri dati, che funziona bene anche con set di dati di immagini relativamente piccoli (il nostro caso).
Quindi, partendo dalle immagini grezze, le ridimensioneremo (96x96) pixel e le invieremo al nostro blocco Transfer Learning:

Pre-elaborazione (Generazione di Feature)
Oltre a ridimensionare le immagini, possiamo cambiarle in scala di grigi o mantenere la profondità di colore RGB effettiva. Iniziamo selezionando `Grayscale`. In questo modo, ognuno dei nostri campioni di dati avrà dimensione 9.216 feature (96x96x1). Con RGB, questa dimensione sarebbe tre volte più grande. Lavorare con la scala di grigi aiuta a ridurre la quantità di memoria finale necessaria per l'inferenza.

Ricordarsi di `[Save parameters]`. Questo genererà le feature da utilizzare nel training.

Progettazione del Modello

Transfer Learning

Nel 2007, Google ha introdotto MobileNetV1, una famiglia di reti neurali di visione artificiale di uso generale progettate pensando ai dispositivi mobili per supportare classificazione, rilevamento e altro. Le MobileNet sono modelli piccoli, a bassa latenza e a basso consumo, parametrizzati per soddisfare i vincoli di risorse di vari casi d'uso.
Sebbene l'architettura di base di MobileNet sia già minuscola e abbia una bassa latenza, spesso un caso d'uso o un'applicazione specifica potrebbe richiedere che il modello sia più piccolo e veloce. MobileNet introduce un parametro semplice α (alfa) chiamato moltiplicatore di larghezza per costruire questi modelli più piccoli e meno costosi dal punto di vista computazionale. Il ruolo del moltiplicatore di larghezza α è di assottigliare una rete in modo uniforme a ogni livello.

Edge Impulse Studio ha **MobileNet V1 (immagini 96x96)** e **V2 (immagini 96x96 e 160x160)** disponibili, con diversi valori **α** (da 0,05 a 1,0). Ad esempio, si otterrà la massima accuratezza con V2, immagini 160x160 e α=1,0. Naturalmente, c'è un compromesso. Maggiore è la precisione, maggiore sarà la memoria (circa 1,3 M di RAM e 2,6 M di ROM) necessaria per eseguire il modello, il che implica una maggiore latenza.

L'ingombro più piccolo sarà ottenuto all'altro estremo con **MobileNet V1** e α=0,10 (circa 53,2 K di RAM e 101 K di ROM).

Per questo primo passaggio, utilizzeremo **MobileNet V1** e α=0,10.

Training

Data Augmentation

Un'altra tecnica necessaria da usare con il deep learning è il **data augmentation**. Il data augmentation è un metodo che può aiutare a migliorare l'accuratezza dei modelli di machine learning, creando dati artificiali aggiuntivi. Un sistema di Data Augmentation apporta piccole modifiche casuali ai dati di training (ad esempio capovolgendo, ritagliando o ruotando le immagini).
Sotto, qui si può vedere come Edge Impulse implementa una policy di data Augmentation sui dati:

```
# Implements the data augmentation policy
def augment_image(image, label):
    # Flips the image randomly
    image = tf.image.random_flip_left_right(image)

    # Increase the image size, then randomly crop it down to
    # the original dimensions
    resize_factor = random.uniform(1, 1.2)
    new_height = math.floor(resize_factor * INPUT_SHAPE[0])
    new_width = math.floor(resize_factor * INPUT_SHAPE[1])
    image = tf.image.resize_with_crop_or_pad(image, new_height, new_width)
    image = tf.image.random_crop(image, size=INPUT_SHAPE)

    # Vary the brightness of the image
    image = tf.image.random_brightness(image, max_delta=0.2)

    return image, label
```

L'esposizione a queste variazioni durante l'addestramento può aiutare a impedire al modello di prendere scorciatoie "memorizzando" indizi superficiali nei dati di addestramento, il che significa che potrebbe riflettere meglio i pattern profondi in esame nel set di dati.
L'ultimo layer del nostro modello avrà 16 neuroni con un dropout del 10% per prevenire l'overfitting. Ecco l'output del Training:

Il risultato potrebbe essere migliore. Il modello ha raggiunto circa il 77% di accuratezza, ma la quantità di RAM prevista per essere utilizzata durante l'inferenza è relativamente piccola (circa 60 KByte), il che è molto buono.

Deployment

Il modello addestrato verrà distribuito come libreria Arduino .zip:

Apri l'IDE Arduino e in **Sketch,** si va su **Include Library** e **add.ZIP Library**. Selezionare il file che scaricato da Edge Impulse Studio e il gioco è fatto!

Nella scheda **Examples** su Arduino IDE, si trova un codice sketch sotto il nome del proprio progetto.

Aprire l'esempio Static Buffer:

La prima riga di codice è esattamente la chiamata di una libreria con tutto il necessario per eseguire l'inferenza sul dispositivo.

```
#include <XIAO-ESP32S3-CAM-Fruits-vs-Veggies_inferencing.h>
```

Ovviamente, questo è un codice generico (un "template") che ottiene solo un campione di dati grezzi (memorizzati nella variabile: features = {} ed esegue il classificatore, eseguendo l'inferenza. Il risultato viene mostrato sul monitor seriale.
Dovremmo ottenere il campione (immagine) dalla fotocamera e pre-elaborarlo (ridimensionandolo a 96x96, convertendolo in scala di grigi e appiattendolo). Questo sarà il tensore di input del nostro modello. Il tensore di output sarà un vettore con tre valori (etichette), che mostrano le probabilità di ciascuna delle classi.

Tornando al progetto (Tab Image), copiare uno dei Raw Data Sample:

9.216 feature verranno copiate negli appunti. Questo è il tensore di input (un'immagine appiattita di 96x96x1), in questo caso, banane. "Incollare" questo tensore di input su

```
features[] = {0xb2d77b, 0xb5d687, 0xd8e8c0, 0xeaecba, 0xc2cf67, ...}
```

Edge Impulse ha incluso la libreria ESP NN nel suo SDK, che contiene funzioni NN (Neural Network) ottimizzate per vari chip Espressif, tra cui ESP32S3 (in esecuzione su Arduino IDE). Quando si esegue l'inferenza, si deve ottenere il punteggio più alto per "banana".

Ottime notizie! Il nostro dispositivo gestisce un'inferenza, scoprendo che l'immagine in ingresso è una banana. Notare inoltre che il tempo di inferenza è stato di circa 317 ms, con un massimo di 3 fps se si è provato a classificare le immagini da un video.
Ora, dovremmo incorporare la telecamera e classificare le immagini in tempo reale.
Si va su Arduino IDE Examples e si scarica dal progetto lo sketch `esp32_camera`:

Si devono cambiare le righe dalla 32 alla 75, che definiscono il modello e i pin della telecamera, utilizzando i dati relativi al nostro modello. Copiare e incollare le righe seguenti, sostituendo le righe 32-75:

```
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
#define Y9_GPIO_NUM       48
#define Y8_GPIO_NUM       11
#define Y7_GPIO_NUM       12
#define Y6_GPIO_NUM       14
#define Y5_GPIO_NUM       16
#define Y4_GPIO_NUM       18
#define Y3_GPIO_NUM       17
#define Y2_GPIO_NUM       15
#define VSYNC_GPIO_NUM    38
#define HREF_GPIO_NUM     47
#define PCLK_GPIO_NUM     13
```

Ecco il codice risultante:

Lo sketch modificato è scaricabile da GitHub: xiao_esp32s3_camera.

> Notare che si possono facoltativamente mantenere i pin come file .h come abbiamo fatto nel Setup Lab.

Caricare il codice sullo XIAO ESP32S3 Sense e si sarà pronti per iniziare a classificare la frutta e la verdura! Si può controllare il risultato su Serial Monitor.

Test del modello (inferenza)

Scattando una foto con la fotocamera, il risultato della classificazione apparirà su Serial Monitor:

Altri test:

Test con un Modello Più Grande

Ora, passiamo all'altro lato delle dimensioni del modello. Selezioniamo un MobilinetV2 96x96 0.35, con immagini RGB in input.

Anche con un modello più grande, la precisione potrebbe essere migliore e la quantità di memoria necessaria per eseguire il modello aumenta di cinque volte, con una latenza che aumenta di sette volte.

> Notare che le prestazioni qui sono stimate con un dispositivo più piccolo, l'ESP-EYE. L'inferenza effettiva con l'ESP32S3 dovrebbe essere migliore.

Per migliorare il nostro modello, dovremo addestrare più immagini.
Anche se il nostro modello non ha migliorato la precisione, testiamo se l'XIAO può gestire un modello così grande. Faremo un semplice test di inferenza con lo sketch Static Buffer. Ridistribuiamo il modello. Se il compilatore EON è abilitato quando si genera la libreria, la memoria totale necessaria per l'inferenza dovrebbe essere ridotta, ma ciò non influisce sulla precisione.

> ⚠️ **Attenzione** - Xiao ESP32S3 con PSRAM abilitata ha memoria sufficiente per eseguire l'inferenza, anche in un modello così grande. Mantenere il Compilatore EON **NOT ENABLED**.

Facendo un'inferenza con MobilinetV2 96x96 0.35, con immagini RGB in input, la latenza è stata di 219 ms, il che è ottimo per un modello così grande.

Per il test, possiamo addestrare di nuovo il modello, usando la versione più piccola di MobileNet V2, con un alpha di 0,05. È interessante che il risultato in accuratezza sia stato più alto.

Notare che la latenza stimata per un Arduino Portenta (o Nicla), in esecuzione con un clock di 480 MHz è di 45 ms.

Distribuendo il modello, abbiamo ottenuto un'inferenza di soli 135 ms, ricordando che XIAO funziona con metà del clock utilizzato da Portenta/Nicla (240 MHz):

Esecuzione dell'inferenza su SenseCraft-Web-Toolkit

Una limitazione significativa della visualizzazione dell'inferenza su Arduino IDE è che non possiamo vedere su cosa punta la telecamera. Una buona alternativa è **SenseCraft-Web-Toolkit**, uno strumento di distribuzione del modello visivo fornito da SSCMA(Seeed SenseCraft Model Assistant). Questo strumento consente di distribuire facilmente modelli su varie piattaforme tramite semplici operazioni. Lo strumento offre un'interfaccia intuitiva e non richiede alcuna codifica.

Seguire i seguenti passaggi per avviare SenseCraft-Web-Toolkit:

1. Aprire il sito web di SenseCraft-Web-Toolkit.
2. Collega XIAO al computer:

- Dopo aver collegato XIAO, selezionarlo come di seguito:

- Selezionare il dispositivo/Porta e premere `[Connect]`:

Si possono provare diversi modelli di Computer Vision caricati in precedenza da Seeed Studio. Da provare e verificarli!

Nel nostro caso, useremo il pulsante blu in fondo alla pagina: `[Upload Custom AI Model]`.

Ma prima, dobbiamo scaricare da Edge Impulse Studio il modello **quantized.tflite**.

3. Si va sul proprio progetto su Edge Impulse Studio, oppure si clona questo:

- XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v1-ESP-NN

4. Nella `Dashboard`, scaricare il modello ("block output"): `Transfer learning model - TensorFlow Lite (int8 quantized)`.

5. Su SenseCraft-Web-Toolkit, usare il pulsante blu in fondo alla pagina: `[Upload Custom AI Model]`. Si aprirà una finestra. Inserire il file del Modello scaricato sul computer da Edge Impulse Studio, scegliere un nome del modello e inserirlo con le etichette (ID: Object):

Notare che si devono usare le etichette addestrate su EI Studio, inserendole in ordine alfabetico (nel nostro caso: apple, banana, potato).

Dopo alcuni secondi (o minuti), il modello verrà caricato sul dispositivo e l'immagine della telecamera apparirà in tempo reale nel Preview Sector:

Il risultato della classificazione sarà in cima all'immagine. Si può anche selezionare la "Confidence" del cursore di inferenza `Confidence`.
Cliccando sul pulsante in alto (Device Log), si può aprire un Serial Monitor per seguire l'inferenza, come fatto con l'IDE Arduino:

Su Device Log, si otterranno informazioni come:

```
perf: {"preprocess":4,"inference":106,"postprocess":0}
classes: [[89,0]]
```

- Tempo di pre-elaborazione (cattura immagine e Crop): 4ms,
- Tempo di inferenza (latenza modello): 106ms,
- Tempo di Post-elaborazione (visualizzazione dell'immagine e inclusione dei dati): 0ms,
- Tensore di output (classi), ad esempio: [[89,0]]; dove 0 è Apple (1 è banana e 2 è patata).

Ecco altri screenshot:

Conclusione

XIAO ESP32S3 Sense è molto flessibile, poco costoso e facile da programmare. Il progetto dimostra il potenziale di TinyML. La memoria non è un problema; il dispositivo può gestire molte attività di post-elaborazione, tra cui la comunicazione.
L'ultima versione del codice si trova nel repository GitHub: XIAO-ESP32S3-Sense.

Risorse

- Codici XIAO ESP32S3
- Dataset
- Progetto Edge Impulse

Rilevamento degli Oggetti

DALL·E prompt - Fumetto in stile anni '50, che mostra una scheda dettagliata con sensori, in particolare una telecamera, su un tavolo con un panno fantasia. Dietro la scheda, un computer con un ampio retro mostra l'IDE Arduino. Il contenuto dell'IDE accenna alle assegnazioni dei pin LED e all'inferenza di apprendimento automatico per rilevare i comandi vocali. Il monitor seriale, in una finestra distinta, rivela gli output per i comandi 'yes' e 'no'.

Introduzione

Nell'ultima sezione riguardante Computer Vision (CV) e XIAO ESP32S3, *Classificazione delle immagini*, abbiamo imparato come impostare e classificare le immagini con questa straordinaria scheda di sviluppo. Continuando il nostro viaggio con CV, esploreremo il **Rilevamento degli oggetti** sui microcontrollori.

Object Detection e Image Classification

Il compito principale con i modelli di Classificazione delle immagini è identificare la categoria di oggetti più probabile presente su un'immagine, ad esempio, per classificare tra un gatto o un cane, gli "oggetti" dominanti in un'immagine:

Ma cosa succede se non c'è una categoria dominante nell'immagine?

```
[PREDICTION]        [Prob]

ashcan              : 27%
Egyptian cat        : 19%
hamper              : 13%
```

Un modello di classificazione delle immagini identifica l'immagine soprastante in modo completamente sbagliato come un "ashcan", probabilmente a causa delle tonalità di colore.

> Il modello utilizzato nelle immagini precedenti è MobileNet, che è addestrato con un ampio set di dati, *ImageNet*, in esecuzione su un Raspberry Pi.

Per risolvere questo problema, abbiamo bisogno di un altro tipo di modello, in cui non solo possono essere trovate **più categorie** (o etichette), ma anche **dove** si trovano gli oggetti in una determinata immagine.

Come possiamo immaginare, tali modelli sono molto più complicati e più grandi, ad esempio, **MobileNetV2 SSD FPN-Lite 320x320, addestrato con il set di dati COCO**. Questo modello di rilevamento degli oggetti pre-addestrato è progettato per individuare fino a 10 oggetti all'interno di un'immagine, generando un riquadro di delimitazione per ogni oggetto rilevato. L'immagine sottostante è il risultato di un tale modello in esecuzione su un Raspberry Pi:

I modelli utilizzati per il rilevamento di oggetti (come MobileNet SSD o YOLO) hanno solitamente dimensioni di diversi MB, il che è OK per l'uso con Raspberry Pi ma non è adatto per l'uso con dispositivi embedded, dove la RAM di solito ha, al massimo, pochi MB come nel caso di XIAO ESP32S3.

Una soluzione innovativa per il Rilevamento di Oggetti: FOMO

Edge Impulse ha lanciato nel 2022, **FOMO** (Faster Objects, More Objects), una nuova soluzione per eseguire il rilevamento di oggetti su dispositivi embedded, come Nicla Vision e Portenta (Cortex M7), su CPU Cortex M4F (serie Arduino Nano33 e OpenMV M4) e sui dispositivi Espressif ESP32 (ESP-CAM, ESP-EYE e XIAO ESP32S3 Sense).
In questo progetto pratico, esploreremo l'Object Detection utilizzando FOMO.

> Per saperne di più sulla FOMO, si può leggere l'annuncio ufficiale su FOMO di Edge Impulse, dove Louis Moreau e Mat Kelcey spiegano in dettaglio come funziona.

Obiettivo del Progetto di Object Detection

Tutti i progetti di apprendimento automatico devono iniziare con un obiettivo dettagliato. Supponiamo di trovarci in una struttura industriale o rurale e di dover smistare e contare **arance (frutti)** e in particolare **rane (insetti)**.

In altre parole, dovremmo eseguire una classificazione multi-etichetta, in cui ogni immagine può avere tre classi:

- Background [Sfondo] (nessun oggetto)
- Fruit
- Bug

Ecco alcuni esempi di immagini non etichettate che dovremmo utilizzare per rilevare gli oggetti (frutti e insetti):

Siamo interessati a quale oggetto è presente nell'immagine, alla sua posizione (centroide) e a quanti ne possiamo trovare su di essa. La dimensione dell'oggetto non viene rilevata con FOMO, come con MobileNet SSD o YOLO, in cui il Bounding Box è uno degli output del modello.
Svilupperemo il progetto utilizzando XIAO ESP32S3 per l'acquisizione di immagini e l'inferenza del modello. Il progetto ML verrà sviluppato utilizzando Edge Impulse Studio. Ma prima di iniziare il progetto di "object detection" in Studio, creiamo un *dataset grezzo* (non etichettato) con immagini che contengono gli oggetti da rilevare.

Raccolta Dati

Si possono catturare immagini usando XIAO, il telefono o altri dispositivi. Qui, useremo XIAO con codice dalla libreria Arduino IDE ESP32.

Raccolta di Dataset con XIAO ESP32S3

Aprire Arduino IDE e selezionare la scheda XIAO_ESP32S3 (e la porta a cui è collegata). Su `File > Examples > ESP32 > Camera`, selezionare `CameraWebServer`.
Nel pannello BOARDS MANAGER, confermare di aver installato l'ultimo pacchetto "stable".

> ⚠ **Attenzione**
>
> Le versioni Alpha (ad esempio, 3.x-alpha) non funzionano correttamente con XIAO ed Edge Impulse. Utilizzare invece l'ultima versione stabile (ad esempio, 2.0.11).

Si devono anche commentare tutti i modelli di fotocamere, eccetto i pin del modello XIAO:
`#define CAMERA_MODEL_XIAO_ESP32S3 // Has PSRAM`
E su `Tools`, abilitare la PSRAM. Inserisci le credenziali wifi e caricare il codice sul dispositivo:

Se il codice viene eseguito correttamente, si vedrà l'indirizzo sul monitor seriale:

Copiare l'indirizzo sul browser e attendere che la pagina venga caricata. Selezionare la risoluzione della telecamera (ad esempio, QVGA) e selezionare `[START STREAM]`. Attendi qualche secondo/minuto, a seconda della connessione. Si può salvare un'immagine nell'area download del computer usando il pulsante [Save].

Edge impulse suggerisce che gli oggetti dovrebbero essere simili per dimensione e non sovrapposti per prestazioni migliori. Questo va bene in una struttura industriale, dove la telecamera dovrebbe essere fissa, mantenendo la stessa distanza dagli oggetti da rilevare. Nonostante ciò, proveremo anche a usare dimensioni e posizioni miste per vedere il risultato.

> Non abbiamo bisogno di creare cartelle separate per le nostre immagini perché ognuna contiene più etichette.

Suggeriamo di usare circa 50 immagini per mescolare gli oggetti e variare il numero di ciascuno che appare sulla scena. Provare ad acquisire con diverse angolazioni, sfondi e condizioni di luce.

> Le immagini archiviate usano una dimensione del frame QVGA di 320x240 e RGB565 (formato pixel colore).

Dopo aver acquisito il dataset, `[Stop Stream]` e spostare le immagini in una cartella.

Edge Impulse Studio

Setup del progetto

Si va su Edge Impulse Studio, si inseriscono le proprie credenziali in **Login** (o si crea un account) e si avvia un nuovo progetto.

Qui, è possibile clonare il progetto sviluppato per questa esercitazione pratica: XIAO-ESP32S3-Sense-Object_Detection

Nella dashboard del progetto, andare in basso e su **Project info** e selezionare **Bounding boxes (object detection)** e **Espressif ESP-EYE** (il più simile alla nostra scheda) come Target Device:

Caricamento dei dati non etichettati

In Studio, si va alla scheda `Data acquisition` e nella sezione `UPLOAD DATA` caricare i file acquisiti come cartella dal computer.

Si può lasciare che Studio divida automaticamente i dati tra “Train” e “Test” o farlo manualmente. Caricheremo tutti come training.

Tutte le immagini non etichettate (47) sono state caricate, ma devono essere etichettate in modo appropriato prima di essere utilizzate come dataset del progetto. Studio ha uno strumento per questo scopo, che che si trova nel link Labeling queue (47).
Ci sono due modi per eseguire l’etichettatura assistita dall’IA su Edge Impulse Studio (versione gratuita):

- Utilizzando yolov5
- Tracciando di oggetti tra i frame

Edge Impulse ha lanciato una funzione di auto-labeling per i clienti Enterprise, semplificando le attività di etichettatura nei progetti di rilevamento degli oggetti.

Gli oggetti ordinari possono essere rapidamente identificati ed etichettati utilizzando una libreria esistente di modelli di rilevamento degli oggetti pre-addestrati da YOLOv5 (addestrati con il set di dati COCO). Ma poiché, nel nostro caso, gli oggetti non fanno parte dei dataset COCO, dovremmo selezionare l'opzione di "tracking" [tracciamento] degli oggetti. Con questa opzione, una volta disegnati i bounding box ed etichettate le immagini in un frame, gli oggetti verranno tracciati automaticamente da un frame all'altro, etichettando *partially* quelli nuovi (non tutti sono etichettati correttamente).

Si può usare EI uploader per importare i dati se si ha già un dataset etichettato contenente dei "bounding box".

Etichettatura del Dataset

Iniziando dalla prima immagine dei dati non etichettati, si usa il mouse per trascinare una casella attorno a un oggetto per aggiungere un'etichetta. Poi si clicca su **Save labels** per passare all'elemento successivo.

Si continua con questo processo finché la coda non è vuota. Alla fine, tutte le immagini dovrebbero avere gli oggetti etichettati come i campioni sottostanti:

Poi, si esaminano i campioni etichettati nella scheda `Data acquisition`. Se una delle etichette è errata, la si può modificarla utilizzando il menù *tre puntini* dopo il nome del campione:

Si verrà guidati a sostituire l'etichetta errata e a correggere il dataset.

Bilanciamento del dataset e suddivisione Train/Test

Dopo aver etichettato tutti i dati, ci siamo resi conto che la classe fruit aveva molti più campioni di bug. Quindi, sono state raccolte 11 immagini di bug nuove e aggiuntive (per un

totale di 58 immagini). Dopo averle etichettate, è il momento di selezionare alcune immagini e spostarle nel dataset di test. Per farlo si usa il menù a tre punti dopo il nome dell'immagine. Sono state selezionate sei immagini, che rappresentano il 13% del set di dati totale.

Impulse Design

In questa fase, si deve definire come:

- Il **Pre-processing** consiste nel ridimensionare le singole immagini da 320 x 240 a 96 x 96 e nel ridurle (forma quadrata, senza ritaglio). In seguito, le immagini vengono convertite da RGB a scala di grigi.
- **Design a Model**, in questo caso, "Object Detection".

Pre-elaborazione di tutti i dataset

In questa sezione, selezionare **Color depth** come Grayscale, adatta per l'uso con modelli FOMO ed eseguire il "Save" dei parametri.

Studio passa automaticamente alla sezione successiva, "Generate features", in cui tutti i campioni verranno pre-elaborati, generando un set di dati con singole immagini 96x96x1 o 9.216 feature.

L'esploratore di feature mostra che tutti i campioni evidenziano una buona separazione dopo la generazione delle feature.

> Alcuni campioni sembrano stare nello spazio sbagliato, ma cliccandoci sopra si conferma la corretta etichettatura.

Progettazione, Addestramento e Test del Modello

Useremo FOMO, un modello di rilevamento degli oggetti basato su MobileNetV2 (alpha 0.35) progettato per segmentare grossolanamente un'immagine in una griglia di **background** rispetto a **oggetti di interesse** (in questo caso, *scatole* e *ruote*).
FOMO è un modello di apprendimento automatico innovativo per il rilevamento degli oggetti, che può utilizzare fino a 30 volte meno energia e memoria rispetto ai modelli tradizionali come Mobilenet SSD e YOLOv5. FOMO può funzionare su microcontrollori con meno di 200 KB di RAM. Il motivo principale per cui ciò è possibile è che mentre altri modelli calcolano le dimensioni dell'oggetto disegnando un quadrato attorno ad esso (bounding box), FOMO ignora le dimensioni dell'immagine, fornendo solo le informazioni su dove si trova l'oggetto nell'immagine tramite le sue coordinate del centroide.

Come funziona FOMO?
FOMO prende l'immagine in scala di grigi e la divide in blocchi di pixel usando un fattore di 8. Per l'input di 96x96, la griglia è 12x12 (96/8=12). Successivamente, FOMO eseguirà un classificatore attraverso ogni blocco di pixel per calcolare la probabilità che ci sia un box o una ruota in ognuno di essi e, successivamente, determinerà le regioni che hanno la più alta probabilità di contenere l'oggetto (se un blocco di pixel non ha oggetti, verrà classificato come *background*). Dalla sovrapposizione della regione finale, FOMO fornisce le coordinate (relative alle dimensioni dell'immagine) del centroide di questa regione.

Per l'addestramento, dovremmo selezionare un modello pre-addestrato. Usiamo **FOMO (Faster Objects, More Objects) MobileNetV2 0.35**. Questo modello utilizza circa 250 KB di RAM e 80 KB di ROM (Flash), che si adatta bene alla nostra scheda.

Per quanto riguarda gli iperparametri di training, il modello verrà addestrato con:

- Epochs: 60
- Batch size: 32
- Learning Rate: 0.001.

Per la convalida durante l'addestramento, il 20% del set di dati (*validation_dataset*) verrà risparmiato. Per il restante 80% (*train_dataset*), applicheremo il "Data Augmentation", che capovolgerà casualmente, cambierà le dimensioni e la luminosità dell'immagine e le ritaglierà, aumentando artificialmente il numero di campioni sul set di dati per l'addestramento.
Di conseguenza, il modello termina con un punteggio F1 complessivo dell'85%, simile al risultato ottenuto utilizzando i dati di prova (83%).

Notare che FOMO ha aggiunto automaticamente una terza etichetta di background [sfondo] ai due precedentemente definiti (*box* e *wheel*).

Nelle attività di rilevamento di oggetti, l'accuratezza non è in genere la metrica di valutazione primaria. Il rilevamento di oggetti comporta la classificazione degli oggetti e la definizione di riquadri di delimitazione attorno a essi, il che lo rende un problema più complesso della semplice classificazione. Il problema è che non abbiamo il riquadro di delimitazione, solo i centroidi. In breve, usare l'accuratezza come metrica potrebbe essere fuorviante e potrebbe non fornire una comprensione completa delle prestazioni del modello. Per questo motivo, useremo il punteggio F1.

Modello di test con "Live Classification"

Una volta addestrato il nostro modello, possiamo testarlo utilizzando lo strumento Live Classification. Nella sezione corrispondente, cliccare sull'icona "Connect a development board" (una piccola MCU) e scansionare il codice QR col telefono.

Una volta connesso, si può usare lo smartphone per catturare immagini reali da testare col modello addestrato su Edge Impulse Studio.

Una cosa da notare è che il modello può produrre falsi positivi e falsi negativi. Questo può essere ridotto al minimo definendo una “Confidence Threshold” appropriata (usare il menù “Tre puntini” per la configurazione). Provare con 0,8 o più.

Deploying del Modello (Arduino IDE)

Selezionare la Libreria Arduino e il modello Quantized (int8), abilitare il compilatore EON nella scheda Deploy e premere [Build].

Apri l’Arduino IDE e, in Sketch, andare su Include Library e aggiungere .ZIP Library. Selezionare il file che scaricato da Edge Impulse Studio e il gioco è fatto!

Nella scheda Examples su Arduino IDE, si trova il codice di uno sketch (`esp32 > esp32_camera`) sotto il nome del progetto.

Si devono cambiare le righe dalla 32 alla 75, che definiscono il modello e i pin della telecamera, utilizzando i dati relativi al nostro modello. Copiare e incollare le righe seguenti, sostituendo le righe 32-75:

```
#define PWDN_GPIO_NUM     -1
#define RESET_GPIO_NUM    -1
#define XCLK_GPIO_NUM     10
#define SIOD_GPIO_NUM     40
#define SIOC_GPIO_NUM     39
#define Y9_GPIO_NUM       48
#define Y8_GPIO_NUM       11
#define Y7_GPIO_NUM       12
#define Y6_GPIO_NUM       14
#define Y5_GPIO_NUM       16
#define Y4_GPIO_NUM       18
#define Y3_GPIO_NUM       17
#define Y2_GPIO_NUM       15
#define VSYNC_GPIO_NUM    38
#define HREF_GPIO_NUM     47
#define PCLK_GPIO_NUM     13
```

Ecco il codice risultante:

Caricare il codice sul XIAO ESP32S3 Sense e si è pronti a iniziare a rilevare frutta e insetti. Si può controllare il risultato su Serial Monitor.

Background

Fruits

Bugs

Si noti che la latenza del modello è di 143ms e il frame rate al secondo è di circa 7 fps (simile a quanto ottenuto con il progetto Image Classification). Ciò accade perché FOMO è intelligentemente costruito su un modello CNN, non con un modello di rilevamento degli oggetti come SSD MobileNet. Ad esempio, quando si esegue un modello MobileNetV2 SSD FPN-Lite 320x320 su un Raspberry Pi 4, la latenza è circa cinque volte superiore (circa 1,5 fps).

Distribuzione del Modello (SenseCraft-Web-Toolkit)

Come discusso nel capitolo Image Classification, verificare l'inferenza con i modelli di immagine su Arduino IDE è molto impegnativo perché non possiamo vedere su cosa punta la telecamera. Di nuovo, utilizziamo **SenseCraft-Web Toolkit**.

Seguire i seguenti passaggi per avviare SenseCraft-Web-Toolkit:

1. Aprire il sito web di SenseCraft-Web-Toolkit.
2. Collega XIAO al computer:

- Dopo aver collegato XIAO, selezionarlo come di seguito:

- Selezionare il dispositivo/Porta e premere `[Connect]`:

Si possono provare diversi modelli di Computer Vision caricati in precedenza da Seeed Studio. Da provare e verificarli!

Nel nostro caso, useremo il pulsante blu in fondo alla pagina: `[Upload Custom AI Model]`.

Ma prima, dobbiamo scaricare da Edge Impulse Studio il modello **quantized .tflite**.

3. Si va sul proprio progetto su Edge Impulse Studio, oppure si clona questo:

- XIAO-ESP32S3-CAM-Fruits-vs-Veggies-v1-ESP-NN

4. Su `Dashboard`, scaricare il modello ("block output"): `Object Detection model - TensorFlow Lite (int8 quantized)`

5. Su SenseCraft-Web-Toolkit, usare il pulsante blu in fondo alla pagina: `[Upload Custom AI Model]`. Si aprirà una finestra. Inserire il file del Modello scaricato sul computer da

Edge Impulse Studio, scegliere un nome del modello e inserirlo con le etichette (ID: Object):

Notare che si devono utilizzare le etichette apprese su EI Studio e inserirle in ordine alfabetico (nel nostro caso, background, bug, fruit).

Dopo alcuni secondi (o minuti), il modello verrà caricato sul dispositivo e l'immagine della telecamera apparirà in tempo reale nel Preview Sector:

Gli oggetti rilevati saranno contrassegnati (il centroide). È possibile selezionare l'affidabilità del cursore di inferenza `Confidence` e `IoU`, che viene utilizzata per valutare l'accuratezza delle "bounding box" previste rispetto a quelle vere.
Cliccando sul pulsante in alto (Device Log), si può aprire un Serial Monitor per seguire l'inferenza, come abbiamo fatto con l'IDE Arduino.

```
perf: {"preprocess":3,"inference":115,"postprocess":1}
boxes: [[30,150,20,20,97,2],[90,110,20,20,98,2],[150,150,20,20,98,2],[170,170,
20,20,94,2],[170,150,20,20,94,2]]
```

Su Device Log, si otterranno informazioni come:

- Tempo di pre-elaborazione (acquisizione dell'immagine e Crop): 3 ms,
- Tempo di inferenza (latenza del modello): 115 ms,
- Tempo di post-elaborazione (visualizzazione dell'immagine e marcatura degli oggetti): 1 ms.
- Tensore di output (box), ad esempio, uno dei box: [[30,150, 20, 20, 97, 2]]; dove 30,150, 20, 20 sono le coordinate della casella (intorno al centroide); 97 è il risultato dell'inferenza e 2 è la classe (in questo caso 2: frutto).

> Notare che nell'esempio precedente, abbiamo ottenuto 5 caselle perché nessuno dei frutti ha ottenuto 3 centroidi. Una soluzione sarà la post-elaborazione, dove possiamo aggregare centroidi vicini in uno.

Ecco altri screenshot:

Conclusione

FOMO è un salto significativo nello spazio di elaborazione delle immagini, come hanno affermato Louis Moreau e Mat Kelcey durante il suo lancio nel 2022:

> FOMO è un algoritmo rivoluzionario che porta per la prima volta il rilevamento, il tracciamento e il conteggio degli oggetti in tempo reale sui microcontrollori.

Esistono molteplici possibilità per esplorare il rilevamento di oggetti (e, più precisamente, il loro conteggio) su dispositivi embedded.

Risorse

- Progetto Edge Impulse

Keyword Spotting (KWS)

Immagine di Marcelo Rovai

Introduzione

Keyword Spotting (KWS) è parte integrante di molti sistemi di riconoscimento vocale, consentendo ai dispositivi di rispondere a parole o frasi specifiche. Sebbene questa tecnologia sia alla base di dispositivi popolari come Google Assistant o Amazon Alexa, è ugualmente applicabile e realizzabile su dispositivi più piccoli e a basso consumo. Questo lab guiderà nell'implementazione di un sistema KWS utilizzando TinyML sulla scheda microcontrollore XIAO ESP32S3.

The XIAO ESP32S3, dotato del chip ESP32-S3 di Espressif, è un microcontrollore compatto e potente che offre un processore Xtensa LX7 dual-core, Wi-Fi integrato e Bluetooth. Il suo equilibrio di potenza di calcolo, efficienza energetica e connettività versatile lo rendono una piattaforma fantastica per le applicazioni TinyML. Inoltre, con la sua scheda di espansione, avremo accesso alla parte "sense" del dispositivo, che ha una fotocamera OV2640 da 1600x1200, uno slot per schede SD e un **microfono digitale**. Il microfono integrato e la scheda SD saranno essenziali in questo progetto.
Utilizzeremo Edge Impulse Studio, una piattaforma potente e intuitiva che semplifica la creazione e l'implementazione di modelli di apprendimento automatico su dispositivi edge. Addestreremo un modello KWS passo dopo passo, ottimizzandolo e distribuendolo su XIAO ESP32S3 Sense.
Il nostro modello sarà progettato per riconoscere parole chiave che possono attivare il "wake-up" [risveglio] del dispositivo o azioni specifiche (nel caso di "YES"), dando vita a progetti con comandi vocali.
Sfruttando la nostra esperienza con TensorFlow Lite per microcontrollori (il motore "sotto il cofano" di EI Studio), creeremo un sistema KWS in grado di apprendere in tempo reale sul dispositivo.
Procedendo nel lab, analizzeremo ogni fase del processo, dalla raccolta e preparazione dei dati all'addestramento e distribuzione del modello, per fornire una comprensione completa dell'implementazione di un sistema KWS su un microcontrollore.

Come funziona un assistente vocale?

Keyword Spotting (KWS) è fondamentale per molti assistenti vocali, consentendo ai dispositivi di rispondere a parole o frasi specifiche. Per iniziare, è essenziale rendersi conto che gli assistenti vocali sul mercato, come Google Home o Amazon Echo-Dot, reagiscono agli umani solo quando vengono "svegliati" da parole chiave specifiche come "Hey Google" sul primo e "Alexa" sul secondo.

In altre parole, il riconoscimento dei comandi vocali si basa su un modello multi-fase o Cascade Detection.

Fase 1: Un microprocessore più piccolo all'interno dell'Echo Dot o Google Home **ascolta continuamente** il suono, in attesa che venga individuata la parola chiave. Per tale rilevamento, viene utilizzato un modello TinyML all'edge (applicazione KWS).

Fase 2: Solo quando vengono attivati dall'applicazione KWS nella Fase 1, i dati vengono inviati al cloud ed elaborati su un modello più grande.
Il video qui sotto mostra un esempio in cui si emula un Google Assistant su un Raspberry Pi (Fase 2), con un Arduino Nano 33 BLE come dispositivo tinyML (Fase 1).

> Per approfondire il progetto completo, guardare il tutorial: Building an Intelligent Voice Assistant From Scratch.

In questo lab, ci concentreremo sulla Fase 1 (KWS o Keyword Spotting), dove utilizzeremo XIAO ESP2S3 Sense, che ha un microfono digitale per individuare la parola chiave.

Il progetto KWS

Il diagramma seguente darà un'idea di come dovrebbe funzionare l'applicazione KWS finale (durante l'inferenza):

La nostra applicazione KWS riconoscerà quattro classi di suono:

- **YES** (Keyword 1)
- **NO** (Keyword 2)
- **NOISE** [rumore] (nessuna parola chiave pronunciata, è presente solo rumore di fondo)
- **UNKNOW** (un mix di parole diverse da YES e NO)

> Facoltativamente, per progetti reali, si consiglia sempre di includere parole diverse dalle parole chiave, come "Rumore" (o Sfondo) e "Sconosciuto".

Il Flusso di Lavoro del Machine Learning

Il componente principale dell'applicazione KWS è il suo modello. Quindi, dobbiamo addestrare un modello del genere con le nostre parole chiave specifiche, rumore e altre parole (lo "unknown"):

Il Dataset

Il componente critico del flusso di lavoro di apprendimento automatico è il **dataset**. Una volta decise le parole chiave specifiche (*YES* e NO), possiamo sfruttare il dataset sviluppato da Pete Warden, "Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition". Questo set di dati ha 35 parole chiave (con +1.000 campioni ciascuna), come yes, no, stop e go. In altre parole, possiamo ottenere 1.500 campioni di *yes* e *no*.
Si può scaricare una piccola parte del dataset da Edge Studio (Keyword spotting pre-built dataset), che include campioni dalle quattro classi che utilizzeremo in questo progetto: yes, no, noise e background. Per farlo, si seguono i passaggi seguenti:

- Download del dataset delle parole chiave.
- Decomprimere il file in una posizione a scelta.

Sebbene disponiamo di molti dati dal dataset di Pete, è consigliabile raccogliere alcune parole pronunciate da noi. Lavorando con gli accelerometri, creare un dataset con dati acquisiti dallo stesso tipo di sensore era essenziale. Nel caso del *suono*, è diverso perché ciò che classificheremo sono, in realtà, dati *audio*.

> La differenza fondamentale tra suono e audio è la loro forma di energia. Il suono è energia delle onde meccaniche (onde sonore longitudinali) che si propagano attraverso un mezzo causando variazioni di pressione all'interno del mezzo. L'audio è costituito da energia elettrica (segnali analogici o digitali) che rappresentano il suono elettricamente.

Le onde sonore dovrebbero essere convertite in dati audio quando pronunciamo una parola chiave. La conversione dovrebbe essere eseguita campionando il segnale generato dal microfono a 16 KHz con una profondità di 16 bit.
Quindi, qualsiasi dispositivo in grado di generare dati audio con questa specifica di base (16Khz/16bit) funzionerà bene. Come dispositivo, possiamo usare il XIAO ESP32S3 Sense appropriato, un computer o persino il telefono cellulare.

Acquisizione di Dati Audio online con Edge Impulse e uno smartphone
Nel lab "Motion Classification" e "Anomaly Detection", colleghiamo il nostro dispositivo direttamente a Edge Impulse Studio per l'acquisizione dei dati (con una frequenza di campionamento da 50 Hz a 100 Hz). Per una frequenza così bassa, potremmo usare la funzione EI CLI *Data Forwarder*, ma secondo Jan Jongboom, CTO di Edge Impulse, *l'audio (*16 KHz)* è troppo veloce perché il data forwarder possa essere acquisito. *Quindi, una volta che i dati digitali sono stati acquisiti dal microfono, possiamo trasformarli* in un file WAV* da inviare a Studio tramite Data Uploader (lo stesso che faremo con il set di dati di Pete).

> Se vogliamo raccogliere dati audio direttamente sullo Studio, possiamo usare qualsiasi smartphone connesso online. Non esploreremo questa opzione qui, ma si può facilmente seguire la documentazione EI.

Acquisizione (offline) di dati audio con XIAO ESP32S3 Sense

Il microfono integrato è il MSM261D3526H1CPM, un microfono MEMS con uscita digitale PDM con Multi-mode. Internamente, è collegato all'ESP32S3 tramite un bus I2S utilizzando i pin IO41 (Clock) e IO41 (Data).

Cos'è I2S?
I2S, o Inter-IC Sound, è un protocollo standard per la trasmissione di audio digitale da un dispositivo a un altro. Inizialmente è stato sviluppato da Philips Semiconductor (ora NXP Semiconductors). È comunemente utilizzato in dispositivi audio come processori di segnale digitale, processori audio digitali e, più di recente, microcontrollori con capacità audio digitali (il nostro caso qui).
Il protocollo I2S è composto da almeno tre linee:

1. Linea di clock di bit (o seriale) (BCLK o CLK): Questa linea si attiva/disattiva per indicare l'inizio di un nuovo bit di dati (pin IO42).

2. Linea di "Word select (WS)": Questa linea si attiva/disattiva per indicare l'inizio di una nuova parola (canale sinistro o canale destro). La frequenza del clock di Word select (WS)

definisce la frequenza di campionamento. Nel nostro caso, L/R sul microfono è impostato su massa, il che significa che utilizzeremo solo il canale sinistro (mono).

3. Data line (SD): Questa linea trasporta i dati audio (pin IO41)
In un flusso di dati I2S, i dati vengono inviati come una sequenza di frame, ciascuno contenente una parola del canale sinistro e una parola del canale destro. Ciò rende I2S particolarmente adatto per la trasmissione di dati audio stereo. Tuttavia, può anche essere utilizzato per audio mono o multicanale con linee dati aggiuntive.
Cominciamo a capire come catturare dati grezzi usando il microfono. Su va sul progetto GitHube si scarica lo sketch: XIAOEsp2s3_Mic_Test:

```
/*
  XIAO ESP32S3 Simple Mic Test
*/

#include <I2S.h>

void setup() {
  Serial.begin(115200);
  while (!Serial) {
  }


  // start I2S at 16 kHz with 16-bits per sample
  I2S.setAllPins(-1, 42, 41, -1, -1);
  if (!I2S.begin(PDM_MONO_MODE, 16000, 16)) {
    Serial.println("Failed to initialize I2S!");
    while (1); // do nothing
  }
}

void loop() {
  // read a sample
  int sample = I2S.read();

  if (sample && sample != -1 && sample != 1) {
    Serial.println(sample);
  }
}
```

Questo codice è un semplice test del microfono per XIAO ESP32S3 che utilizza l'interfaccia I2S (Inter-IC Sound). Imposta l'interfaccia I2S per catturare dati audio a una frequenza di campionamento di 16 kHz con 16 bit per campione e quindi legge continuamente i campioni dal microfono e li stampa sul monitor seriale.
Analizziamo le parti principali del codice:

- Include la libreria I2S: Questa libreria fornisce funzioni per configurare e utilizzare l'interfaccia I2S, che è uno standard per la connessione di dispositivi audio digitali.
- I2S.setAllPins(-1, 42, 41, -1, -1): Imposta i pin I2S. I parametri sono (-1, 42, 41, -1, -1), dove il secondo parametro (42) è il PIN per il clock I2S (CLK) e il terzo parametro (41) è il PIN per la linea dati I2S (DATA). Gli altri parametri sono impostati su -1, il che significa che quei pin non vengono utilizzati.
- I2S.begin(PDM_MONO_MODE, 16000, 16): Inizializza l'interfaccia I2S in modalità mono Pulse Density Modulation (PDM), con una frequenza di campionamento di 16 kHz e 16 bit per campione. Se l'inizializzazione fallisce, viene stampato un messaggio di errore e il programma si arresta.
- int sample = I2S.read(): Legge un campione audio dall'interfaccia I2S.

Se il campione è valido, viene stampato sul monitor seriale e sul plotter.
Di seguito è riportato un test "sussurrato" in due toni diversi.

Salvare campioni audio registrati (dataset) come file audio .wav su una scheda microSD

Utilizziamo il lettore di schede SD integrato per salvare i file audio .wav; dobbiamo prima abilitare la PSRAM XIAO.

> ESP32-S3 ha solo poche centinaia di kilobyte di RAM interna sul chip MCU. Potrebbe essere insufficiente per alcuni scopi, quindi ESP32-S3 può utilizzare fino a 16 MB di PSRAM esterna (Psuedostatic RAM) collegata in parallelo con il chip flash SPI. La memoria esterna è incorporata nella mappa di memoria e, con alcune restrizioni, è utilizzabile allo stesso modo della RAM dati interna.

Per iniziare, si inserisce la scheda SD sullo XIAO come mostrato nella foto qui sotto (la scheda SD deve essere formattata in FAT32).

Attivare la funzione PSRAM del chip ESP-32 (Arduino IDE): Tools>PSRAM: “OPI PSRAM”>OPI PSRAM

- Scaricare lo sketch Wav_Record_dataset,https://github.com/Mjrovai/XIAO-ESP32S3-Sense/tree/main/Wav_Record_dataset che si trova sul GitHub del progetto.

Questo codice registra l’audio usando l’interfaccia I2S della scheda Seeed XIAO ESP32S3 Sense, salva la registrazione come file .wav su una scheda SD e consente il controllo del processo di registrazione tramite comandi inviati dal monitor seriale. Il nome del file audio è personalizzabile (dovrebbe essere le etichette della classe da usare con la formazione) e possono essere effettuate più registrazioni, ciascuna salvata in un nuovo file. Il codice include anche funzionalità per aumentare il volume delle registrazioni.

Analizziamo le parti più essenziali:

```
#include <I2S.h>
#include "FS.h"
#include "SD.h"
#include "SPI.h"
```

Queste sono le librerie necessarie per il programma. I2S.h consente l’input audio, FS.h fornisce capacità di gestione del file system, SD.h consente al programma di interagire con una scheda SD e SPI.h gestisce la comunicazione SPI con la scheda SD.

```
#define RECORD_TIME   10
#define SAMPLE_RATE 16000U
#define SAMPLE_BITS 16
#define WAV_HEADER_SIZE 44
#define VOLUME_GAIN 2
```

Qui vengono definite varie costanti per il programma.

- **RECORD_TIME** specifica la lunghezza della registrazione audio in secondi.
- **SAMPLE_RATE** e **SAMPLE_BITS** definiscono la qualità audio della registrazione.
- **WAV_HEADER_SIZE** specifica la dimensione dell’intestazione del file .wav.

- **VOLUME_GAIN** viene utilizzato per aumentare il volume della registrazione.

```
int fileNumber = 1;
String baseFileName;
bool isRecording = false;
```

Queste variabili tengono traccia del numero di file corrente (per creare nomi di file univoci), del nome del file di base e se il sistema sta attualmente registrando.

```
void setup() {
  Serial.begin(115200);
  while (!Serial);

  I2S.setAllPins(-1, 42, 41, -1, -1);
  if (!I2S.begin(PDM_MONO_MODE, SAMPLE_RATE, SAMPLE_BITS)) {
    Serial.println("Failed to initialize I2S!");
    while (1);
  }


  if(!SD.begin(21)){
    Serial.println("Failed to mount SD Card!");
    while (1);
  }

  Serial.printf("Enter with the label name\n");
}
```

La funzione di configurazione inizializza la comunicazione seriale, l'interfaccia I2S per l'ingresso audio e l'interfaccia della scheda SD. Se l'I2S non si inizializza o la scheda SD non riesce a essere montata, verrà visualizzato un messaggio di errore e l'esecuzione verrà interrotta.

```
void loop() {
  if (Serial.available() > 0) {
    String command = Serial.readStringUntil('\n');
    command.trim();
    if (command == "rec") {
      isRecording = true;
    } else {
      baseFileName = command;
      fileNumber = 1; //reset file number each time a new basefile name is set
      Serial.printf("Send rec for starting recording label \n");
    }

  }

  if (isRecording && baseFileName != "") {
    String fileName = "/" + baseFileName + "." + String(fileNumber) + ".wav";
    fileNumber++;
    record_wav(fileName);
    delay(1000); // delay to avoid recording multiple files at once
    isRecording = false;
  }
}
```

Nel ciclo principale, il programma attende un comando dal monitor seriale. Se il comando è rec, il programma inizia a registrare. Altrimenti, si presume che il comando sia il nome base per i file .wav. Se sta attualmente registrando e un nome file base è impostato, registra l'audio e lo salva come file .wav. I nomi dei file vengono generati aggiungendo il numero al nome file base.

```
void record_wav(String fileName)
{
  ...

  File file = SD.open(fileName.c_str(), FILE_WRITE);
  ...
  rec_buffer = (uint8_t *)ps_malloc(record_size);
  ...

  esp_i2s::i2s_read(esp_i2s::I2S_NUM_0,
```

```
                    rec_buffer,
                    record_size,
                    &sample_size,
                    portMAX_DELAY);
  ...
}
```

Questa funzione registra l'audio e lo salva come file .wav con il nome specificato. Inizia inizializzando le variabili sample_size e record_size. record_size viene calcolato in base alla frequenza di campionamento, alla dimensione e al tempo di registrazione desiderato. Analizziamo le sezioni essenziali;

```
File file = SD.open(fileName.c_str(), FILE_WRITE);
// Write the header to the WAV file
uint8_t wav_header[WAV_HEADER_SIZE];
generate_wav_header(wav_header, record_size, SAMPLE_RATE);
file.write(wav_header, WAV_HEADER_SIZE);
```

Questa sezione del codice apre il file sulla scheda SD per la scrittura e poi genera l'intestazione del file .wav utilizzando la funzione generate_wav_header. Quindi scrive l'intestazione nel file.

```
// PSRAM malloc for recording
rec_buffer = (uint8_t *)ps_malloc(record_size);
if (rec_buffer == NULL) {
  Serial.printf("malloc failed!\n");
  while(1) ;
}
Serial.printf("Buffer: %d bytes\n", ESP.getPsramSize() - ESP.getFreePsram());
```

La funzione ps_malloc alloca memoria nella PSRAM per la registrazione. Se l'allocazione fallisce (ad esempio, rec_buffer è NULL), stampa un messaggio di errore e interrompe l'esecuzione.

```
// Start recording
esp_i2s::i2s_read(esp_i2s::I2S_NUM_0,
         rec_buffer,
         record_size,
         &sample_size,
         portMAX_DELAY);
if (sample_size == 0) {
  Serial.printf("Record Failed!\n");
} else {
    Serial.printf("Record %d bytes\n", sample_size);
  }
```

La funzione i2s_read legge i dati audio dal microfono in rec_buffer. Stampa un messaggio di errore se non vengono letti dati (sample_size è 0).

```
// Increase volume
for (uint32_t i = 0; i < sample_size; i += SAMPLE_BITS/8) {
  (*(uint16_t *)(rec_buffer+i)) <<= VOLUME_GAIN;
}
```

Questa sezione del codice aumenta il volume di registrazione spostando i valori del campione di VOLUME_GAIN.

```
// Write data to the WAV file
Serial.printf("Writing to the file ...\n");
if (file.write(rec_buffer, record_size) != record_size)
  Serial.printf("Write file Failed!\n");

free(rec_buffer);
file.close();
Serial.printf("Recording complete: \n");
Serial.printf("Send rec for a new sample or enter a new label\n\n");
```

Infine, i dati audio vengono scritti nel file .wav. Se l'operazione di scrittura fallisce, viene stampato un messaggio di errore. Dopo la scrittura, la memoria allocata per rec_buffer viene liberata e il file viene chiuso. La funzione termina stampando un messaggio di completamento e chiedendo all'utente di inviare un nuovo comando.

```
void generate_wav_header(uint8_t *wav_header,
             uint32_t wav_size,
```

```
              uint32_t sample_rate)
{
  ...
  memcpy(wav_header, set_wav_header, sizeof(set_wav_header));
}
```

La funzione generate_wav_header crea un'intestazione di file .wav in base ai parametri (wav_size e sample_rate). Genera un array di byte in base al formato di file .wav, che include campi per la dimensione del file, il formato audio, il numero di canali, la frequenza di campionamento, la frequenza di byte, l'allineamento dei blocchi, i bit per campione e la dimensione dei dati. L'intestazione generata viene poi copiata nell'array wav_header passato alla funzione.
Ora, caricare il codice su XIAO e ottenere campioni dalle parole chiave (yes e no). Si possono anche catturare rumore e altre parole.
Il monitor seriale chiederà di ricevere l'etichetta da registrare.

Invia l'etichetta (ad esempio, yes). Il programma attenderà un altro comando: rec

E il programma inizierà a registrare nuovi campioni ogni volta che viene inviato un comando rec. I file verranno salvati come yes.1.wav, yes.2.wav, yes.3.wav, ecc., finché non verrà inviata una nuova etichetta (ad esempio, no). In questo caso, si deve inviare il comando rec per ogni nuovo campione, che verrà salvato come no.1.wav, no.2.wav, no.3.wav, ecc.

Alla fine, otterremo i file salvati sulla scheda SD.

I file sono pronti per essere caricati su Edge Impulse Studio

App di Acquisizione dei Dati Audio (offline)

In alternativa, si può anche usare il PC o lo smartphone per acquisire dati audio con una frequenza di campionamento di 16 KHz e una profondità di bit di 16 bit. Una buona app per questo è *Voice Recorder Pro* (IOS). Le registrazioni si devono salvare come file .wav e inviarle al computer.

Notare che qualsiasi app, come Audacity, può essere usata per la registrazione audio o anche il computer.

Modello di training con Edge Impulse Studio

Caricamento dei Dati

Quando il dataset grezzo è definito e raccolto (dataset di Pete + parole chiave registrate), dovremmo avviare un nuovo progetto in Edge Impulse Studio:

Una volta creato il progetto, selezionare lo strumento "Upload Existing Data" nella sezione "Acquisition section". Si scelgono i file da caricare:

E si caricano nello Studio (si possono dividere automaticamente i dati in train/test). Ripetere per tutte le classi e tutti i dati grezzi.

I campioni appariranno ora nella sezione “Data acquisition”.

Tutti i dati sul dataset di Pete hanno una lunghezza di 1s, ma i campioni registrati nella sezione precedente hanno 10s e devono essere divisi in campioni da 1s per essere compatibili.
Cliccare sui tre punti dopo il nome del campione e selezionare “Split sample”.

Una volta all’interno dello strumento, dividere i dati in record da 1 secondo. Se necessario, aggiungere o rimuovere segmenti:

Questa procedura deve essere ripetuta per tutti i campioni.

> Nota: Per file audio più lunghi (minuti), prima si dividono in segmenti da 10 secondi e poi usa di nuovo lo strumento per ottenere le divisioni finali da 1 secondo.

Supponiamo di non dividere automaticamente i dati in train/test durante il caricamento. In tal caso, possiamo farlo manualmente (utilizzando il menù a tre punti, spostando i campioni singolarmente) o utilizzando Perform Train / Test Split su Dashboard - Danger Zone.

> Possiamo facoltativamente controllare tutti i dataset utilizzando la scheda Data Explorer.

Creazione di Impulse (Pre-Process / Definizione del Modello)

Un **impulse** *prende dati grezzi, usa l'elaborazione del segnale per estrarre le feature e poi usa un blocco di apprendimento per classificare nuovi dati.*

Per prima cosa, prenderemo i dati con una finestra di 1 secondo, aumentando i dati, facendo scorrere quella finestra ogni 500 ms. Notare che è impostata l'opzione Zero-pad data. È essenziale riempire con zeri i campioni inferiori a 1 secondo (in alcuni casi, si è ridotta la finestra di 1000 ms sullo strumento di divisione per evitare rumori e picchi).
Ogni campione audio di 1 secondo dovrebbe essere pre-elaborato e convertito in un'immagine (ad esempio, 13 x 49 x 1). Useremo MFCC, che estrae le caratteristiche dai segnali audio usando Mel Frequency Cepstral Coefficients, che sono ottimi per la voce umana.

Successivamente, selezioniamo KERAS per la classificazione e costruiamo il nostro modello da zero eseguendo la classificazione delle immagini tramite la rete neurale convoluzionale).

Pre-elaborazione (MFCC)

Il passo successivo è creare le immagini da addestrare nella fase successiva: Possiamo mantenere i valori dei parametri di default o sfruttare l'opzione DSP Autotuneparameters, cosa che faremo.

Il risultato non impiegherà molta memoria per pre-elaborare i dati (solo 16 KB). Tuttavia, il tempo di elaborazione stimato è elevato, 675 ms per un Espressif ESP-EYE (il riferimento più vicino disponibile), con un clock di 240 KHz (lo stesso del nostro dispositivo), ma con una CPU più piccola (XTensa LX6, rispetto alla LX7 sull'ESP32S). Il tempo di inferenza reale dovrebbe essere inferiore.
Supponiamo di dover ridurre il tempo di inferenza in seguito. In tal caso, dovremmo tornare alla fase di pre-elaborazione e, ad esempio, ridurre la lunghezza FFT a 256, modificare il numero di coefficienti o un altro parametro.
Per ora, manteniamo i parametri definiti dallo strumento Autotuning. Salviamo i parametri e generiamo le funzionalità.

Per andare oltre con la conversione di dati seriali temporali in immagini usando FFT, spettrogramma, ecc., si può giocare con questo CoLab: Audio Raw Data Analysis.

Progettazione e Addestramento del Modello

Useremo un modello di Rete Neurale Convoluzionale (CNN). L'architettura di base è definita con due blocchi di Conv1D + MaxPooling (rispettivamente con 8 e 16 neuroni) e un Dropout di 0,25. E sull'ultimo layer, dopo aver appiattito quattro neuroni, uno per ogni classe:

Come iperparametri, avremo un Learning Rate di 0,005 e un modello che verrà addestrato per 100 epoche. Includeremo anche l'aumento dei dati, come un po' di rumore. Il risultato sembra OK:

Per capire cosa sta succedendo "sotto il cofano", si può scaricare il dataset ed eseguire un Jupyter Notebook giocando con il codice. Ad esempio, si può analizzare l'accuratezza per ogni epoca:

Questo CoLab Notebook può spiegare come si può andare oltre: KWS Classifier Project - Looking "Under the hood
Training/xiao_esp32s3_keyword_spotting_project_nn_classifier.ipynb)".

Test

Testando il modello con i dati messi da parte prima dell'addestramento (Test Data), abbiamo ottenuto un'accuratezza di circa l'87%.

Model testing results

ACCURACY
86.73%

	NO	NOISE	UNKNOWN	YES	UNCERTAIN
NO	86.3%	0.7%	3.9%	1.4%	7.7%
NOISE	0%	88.6%	3.3%	0.7%	7.5%
UNKNOWN	4.4%	2.7%	78.1%	1.7%	13.1%
YES	0.3%	0%	0.7%	93.9%	5.1%
F1 SCORE	0.90	0.92	0.84	0.95	

Feature explorer

Esaminando il punteggio F1, possiamo vedere che per YES abbiamo ottenuto 0.95, un risultato eccellente una volta utilizzata questa parola chiave per “attivare” la nostra fase di post-elaborazione (accendere il LED integrato). Anche per NO, abbiamo ottenuto 0,90. Il risultato peggiore è per unknown, che è OK.
Possiamo procedere con il progetto, ma è possibile eseguire Live Classification utilizzando uno smartphone prima della distribuzione sul nostro dispositivo. Si va alla sezione Live Classification e si clicca su Connect a Development board:

Puntare il telefono sul codice a barre e selezionare il link.

Il telefono sarà connesso allo Studio. Selezionare l'opzione Classification sull'app e, quando è in esecuzione, iniziare a testare le parole chiave, confermando che il modello funziona con dati live e reali:

Distribuzione e Inferenza

Studio impacchetterà tutte le librerie necessarie, le funzioni di pre-elaborazione e i modelli addestrati, scaricandoli sul computer. Si deve selezionare l'opzione Arduino Library e, in basso, scegliere Quantized (Int8) e premere il pulsante Build.

Ora è il momento di un vero test. Faremo inferenze completamente scollegate da Studio. Modifichiamo uno degli esempi di codice ESP32 creati quando si distribuisce la libreria Arduino.

Nell'IDE Arduino, si va alla scheda File/Examples, si cerca il progetto e si seleziona esp32/esp32_microphone:

Questo codice è stato creato per il microfono integrato ESP-EYE, che dovrebbe essere adattato al nostro dispositivo.

Iniziare a modificare le librerie per gestire il bus I2S:

```
41 /* Includes ---------------------------------
42 #include <XIAO-ESP32S3-KWS_inferencing.h>
43
44 #include "freertos/FreeRTOS.h"
45 #include "freertos/task.h"
46
47 #include "driver/i2s.h"
48
```

Da:

```
#include <I2S.h>
#define SAMPLE_RATE 16000U
#define SAMPLE_BITS 16
```

Inizializzare il microfono IS2 in setup(), includendo le righe:

```
void setup()
{
...
    I2S.setAllPins(-1, 42, 41, -1, -1);
    if (!I2S.begin(PDM_MONO_MODE, SAMPLE_RATE, SAMPLE_BITS)) {
      Serial.println("Failed to initialize I2S!");
    while (1) ;
...
}
```

Nella funzione static void capture_samples(void* arg), sostituire la riga 153 che legge i dati dal microfono I2S:

```
145 static void capture_samples(void* arg) {
146
147   const int32_t i2s_bytes_to_read = (uint32_t)arg;
148   size_t bytes_read = i2s_bytes_to_read;
149
150   while (record_status) {
151
152     /* read data at once from i2s */
153     i2s_read((i2s_port_t)1, (void*)sampleBuffer, i2s_bytes_to_read, &bytes_read, 100);
154
```

Da:

```
/* read data at once from i2s */
esp_i2s::i2s_read(esp_i2s::I2S_NUM_0,
                  (void*)sampleBuffer,
                  i2s_bytes_to_read,
                  &bytes_read, 100);
```

Nella funzione static bool microphone_inference_start(uint32_t n_samples), dovremmo commentare o eliminare le righe da 198 a 200, dove viene chiamata la funzione di inizializzazione del microfono. Ciò non è necessario perché il microfono I2S è già stato inizializzato durante setup().

```
186 static bool microphone_inference_start(uint32_t n_samples)
187 {
188     inference.buffer = (int16_t *)malloc(n_samples * sizeof(int16_t));
189
190     if(inference.buffer == NULL) {
191         return false;
192     }
193
194     inference.buf_count  = 0;
195     inference.n_samples  = n_samples;
196     inference.buf_ready  = 0;
197
198 //    if (i2s_init(EI_CLASSIFIER_FREQUENCY)) {
199 //        ei_printf("Failed to start I2S!");
200 //    }
201
```

Infine, nella funzione static void microphone_inference_end(void), sostituire la riga 243:

```
241 static void microphone_inference_end(void)
242 {
243     i2s_deinit();
244     ei_free(inference.buffer);
245 }
```

Da:

```
static void microphone_inference_end(void)
{
    free(sampleBuffer);
    ei_free(inference.buffer);
}
```

Il codice completo si trova tra i progetti GitHub. Caricare lo sketch sulla bacheca e provare alcune inferenze reali:

```
/dev/cu.usbmodem1101                                                    Send
11:23:32.382 -> Edge Impulse Inferencing Demo
11:23:32.382 -> Inferencing settings:
11:23:32.382 ->         Interval: 0.062500 ms.
11:23:32.382 ->         Frame size: 16000
11:23:32.382 ->         Sample length: 1000 ms.
11:23:32.382 ->         No. of classes: 4
11:23:32.382 ->
11:23:32.382 -> Starting continious inference in 2 seconds...
11:23:34.464 -> Recording...
11:23:35.977 -> Predictions (DSP: 515 ms., Classification: 3 ms., Anomaly: 0 ms.):
11:23:35.977 ->     no: 0.007813
11:23:35.977 ->     noise: 0.964844
11:23:35.977 ->     unknown: 0.023437
11:23:35.977 ->     yes: 0.000000
11:23:36.955 -> Predictions (DSP: 514 ms., Classification: 3 ms., Anomaly: 0 ms.):
11:23:36.955 ->     no: 0.003906
11:23:36.955 ->     noise: 0.957031
11:23:36.955 ->     unknown: 0.015625
11:23:36.955 ->     yes: 0.023437
Autoscroll  Show timestamp        Both NL & CR   115200 baud   Clear output
```

Post-elaborazione

Ora che sappiamo che il modello funziona rilevando le nostre parole chiave, modifichiamo il codice per vedere il LED interno accendersi ogni volta che viene rilevato uno YES.
Si deve inizializzare il LED:

```
#define LED_BUILT_IN 21
...
void setup()
{
...
  pinMode(LED_BUILT_IN, OUTPUT); // Set the pin as output
  digitalWrite(LED_BUILT_IN, HIGH); //Turn off
...
}
```

E modifica la parte "// print the predictions" del codice precedente (su loop():

```
int pred_index = 0;     // Initialize pred_index
float pred_value = 0;   // Initialize pred_value

// print the predictions
ei_printf("Predictions ");
ei_printf("(DSP: %d ms., Classification: %d ms., Anomaly: %d ms.)",
     result.timing.dsp, result.timing.classification, result.timing.anomaly);
ei_printf(": \n");
for (size_t ix = 0; ix < EI_CLASSIFIER_LABEL_COUNT; ix++) {
      ei_printf("    %s: ", result.classification[ix].label);
      ei_printf_float(result.classification[ix].value);
      ei_printf("\n");

      if (result.classification[ix].value > pred_value){
         pred_index = ix;
         pred_value = result.classification[ix].value;
      }
}

// show the inference result on LED
if (pred_index == 3){
    digitalWrite(LED_BUILT_IN, LOW); //Turn on
}
else{
   digitalWrite(LED_BUILT_IN, HIGH); //Turn off
}
```

Il codice completo si trova sul GitHub del progetto. Caricare lo sketch sulla scheda e provare alcune inferenze reali:

L'idea è che il LED sarà ACCESO ogni volta che viene rilevata la parola chiave YES. Allo stesso modo, invece di accendere un LED, questo potrebbe essere un "trigger" per un dispositivo esterno, come abbiamo visto nell'introduzione.

Conclusione

The Seeed XIAO ESP32S3 Sense è un *tiny device gigante*! Tuttavia, è potente, affidabile, non costoso, a basso consumo e ha sensori adatti per essere utilizzato nelle applicazioni di apprendimento automatico embedded più comuni come visione e suono. Anche se Edge Impulse non supporta ufficialmente XIAO ESP32S3 Sense (ancora!), ci siamo resi conto che utilizzare Studio per la formazione e l'implementazione è semplice.

> Nel repository GitHub, si trova l'ultima versione di tutto il codice utilizzato in questo progetto e nei precedenti della serie XIAO ESP32S3.

Prima di concludere, considerare che la classificazione dei suoni è più di una semplice voce. Ad esempio, si possono sviluppare progetti TinyML sul suono in diverse aree, come:

- **Sicurezza** (Rilevamento vetri rotti)
- **Industria** (Rilevamento di Anomalie)
- **Medicina** (Russare, Tosse, Malattie polmonari)
- **Natura** (Controllo alveari, suono degli insetti)

Risorse

- Codici XIAO ESP32S3
- Sottoinsieme del Dataset dei Comandi Vocali di Google
- KWS MFCC Analysis Colab Notebook
- KWS CNN training Colab Notebook
- XIAO ESP32S3 Post-processing Code
- Progetto Edge Impulse

Classificazione del Movimento e Rilevamento delle Anomalie

DALL·E prompt - Illustrazione in stile cartoon anni '50 ambientata in un laboratorio audio d'epoca. Scienziati, vestiti in abiti classici con camici bianchi, analizzano attentamente i dati audio su grandi lavagne. Le lavagne mostrano intricati grafici FFT (Fast Fourier Transform) e curve del dominio del tempo. Sono sparse in giro apparecchiature audio antiche, ma le rappresentazioni dei dati sono chiare e dettagliate, a indicare la loro attenzione all'analisi audio.

Introduzione

The XIAO ESP32S3 Sense, con la sua fotocamera e microfono integrati, è un dispositivo versatile. Ma cosa succede se c'è bisogno di aggiungere un altro tipo di sensore, come un IMU? Nessun problema! Una delle caratteristiche distintive di XIAO ESP32S3 sono i suoi molteplici pin che possono essere utilizzati come bus I2C (pin SDA/SCL), rendendolo una piattaforma adatta per l'integrazione di sensori.

Installazione dell'IMU

Quando si seleziona l'IMU, il mercato offre un'ampia gamma di dispositivi, ognuno con caratteristiche e capacità uniche. Si potrebbe scegliere, ad esempio, ADXL362 (3 assi), MAX21100 (6 assi), MPU6050 (6 assi), LIS3DHTR (3 assi) o LCM20600Seeed Grove— (6 assi), che fa parte dell'IMU 9DOF (lcm20600+AK09918). Questa varietà consente di adattare la scelta alle esigenze specifiche del progetto.
Per questo progetto, utilizzeremo un'IMU, la MPU6050 (o 6500), un'unità accelerometro/giroscopio a 6 assi a basso costo (meno di 2,00 USD).

Alla fine del lab, commenteremo anche l'utilizzo dell'LCM20600.

MPU-6500 è un dispositivo di Motion Tracking a 6 assi che combina un giroscopio a 3 assi, un accelerometro a 3 assi e un Digital Motion ProcessorTM (DMP) in un piccolo package da 3x3x0,9 mm. È inoltre dotato di un FIFO da 4096 byte che può ridurre il traffico sull'interfaccia del bus seriale e ridurre il consumo energetico consentendo al processore di sistema di leggere i dati del sensore in modalità burst e quindi passare a una modalità a basso consumo.
Con il suo bus I2C dedicato al sensore, MPU-6500 accetta direttamente input da dispositivi I2C esterni. MPU-6500, con la sua integrazione a 6 assi, DMP on-chip e firmware di calibrazione runtime, consente ai produttori di eliminare la selezione, la qualificazione e l'integrazione a livello di sistema costose e complesse di dispositivi discreti, garantendo prestazioni di movimento ottimali per i consumatori. MPU-6500 è progettato anche per interfacciarsi con più sensori digitali non inerziali, come i sensori di pressione, sulla sua porta I2C ausiliaria.

MPU6050

MPU6500

Di solito, le librerie disponibili sono per MPU6050, ma funzionano per entrambi i dispositivi.

Collegamento dell'HW

Collegare l'IMU allo XIAO secondo lo schema seguente:

- MPU6050 **SCL** –> XIAO **D5**
- MPU6050 **SDA** –> XIAO **D4**
- MPU6050 **VCC** –> XIAO **3.3V**
- MPU6050 **GND** –> XIAO **GND**

Installare la libreria

Si va su Arduino Library Manager e si digita MPU6050. Installare la versione più recente.

Scaricare lo sketch MPU6050_Acc_Data_Acquisition.in:

```
/*
 * Based on I2C device class (I2Cdev) Arduino sketch for MPU6050 class
   by Jeff Rowberg <jeff@rowberg.net>
 * and Edge Impulse Data Forwarder Exampe (Arduino)
   - https://docs.edgeimpulse.com/docs/cli-data-forwarder
 *
 * Developed by M.Rovai @11May23
 */

#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"

#define FREQUENCY_HZ        50
#define INTERVAL_MS         (1000 / (FREQUENCY_HZ + 1))
#define ACC_RANGE           1 // 0: -/+2G; 1: +/-4G

// convert factor g to m/s^2^ ==> [-32768, +32767] ==> [-2g, +2g]
#define CONVERT_G_TO_MS2    (9.81/(16384.0/(1.+ACC_RANGE)))

static unsigned long last_interval_ms = 0;

MPU6050 imu;
int16_t ax, ay, az;

void setup() {
    Serial.begin(115200);


    // initialize device
    Serial.println("Initializing I2C devices...");
    Wire.begin();
    imu.initialize();
    delay(10);

//    // verify connection
//    if (imu.testConnection()) {
//      Serial.println("IMU connected");
//    }
//    else {
//      Serial.println("IMU Error");
//    }
    delay(300);

    //Set MCU 6050 OffSet Calibration
    imu.setXAccelOffset(-4732);
    imu.setYAccelOffset(4703);
    imu.setZAccelOffset(8867);
    imu.setXGyroOffset(61);
    imu.setYGyroOffset(-73);
    imu.setZGyroOffset(35);

    /* Set full-scale accelerometer range.
     * 0 = +/- 2g
     * 1 = +/- 4g
     * 2 = +/- 8g
     * 3 = +/- 16g
     */
    imu.setFullScaleAccelRange(ACC_RANGE);
}

void loop() {
      if (millis() > last_interval_ms + INTERVAL_MS) {
        last_interval_ms = millis();

        // read raw accel/gyro measurements from device
        imu.getAcceleration(&ax, &ay, &az);

        // converting to m/s^2^
        float ax_m_s^2^ = ax * CONVERT_G_TO_MS2;
```

```
        float ay_m_s^2^ = ay * CONVERT_G_TO_MS2;
        float az_m_s^2^ = az * CONVERT_G_TO_MS2;

        Serial.print(ax_m_s^2^);
        Serial.print("\t");
        Serial.print(ay_m_s^2^);
        Serial.print("\t");
        Serial.println(az_m_s^2^);
      }
}
```

Alcuni commenti sul codice:

Notare che i valori generati dall'accelerometro e dal giroscopio hanno un intervallo: [-32768, +32767], quindi, ad esempio, se viene utilizzato l'intervallo di default dell'accelerometro, l'intervallo in G dovrebbe essere: [-2g, +2g]. Quindi, "1G" significa 16384.

Per la conversione in m/s^2, ad esempio, si può definire quanto segue:

```
#define CONVERT_G_TO_MS2 (9.81/16384.0)
```

Nel codice, è stata lasciata un'opzione (ACC_RANGE) da impostare su 0 (+/-2G) o 1 (+/- 4G). Useremo +/-4G; dovrebbe bastare per noi. In questo caso.

Acquisiremo i dati dell'accelerometro a una frequenza di 50 Hz e i dati di accelerazione saranno inviati alla porta seriale come metri al secondo quadrato (m/s^2).

Quando si esegue il codice con l'IMU appoggiata sul tavolo, i dati dell'accelerometro mostrati sul monitor seriale dovrebbero essere circa 0.00, 0.00 e 9.81. Se i valori sono molto diversi, si deve calibrare l'IMU.

L'MCU6050 può essere calibrato usando lo sketch: mcu6050-calibration.ino.

Eseguire il codice. Sul Serial Monitor verrà visualizzato quanto segue:

Inviare un carattere qualsiasi (nell'esempio sopra, "x") e la calibrazione dovrebbe iniziare.

> Notare che c'è un messaggio MPU6050 di connessione fallita. Ignorare questo messaggio. Per qualche motivo, imu.testConnection() non restituisce un risultato corretto.

Alla fine, si riceveranno i valori di offset da utilizzare su tutti gli sketch:

Prendere i valori e usarli nella configurazione:

```
//Set MCU 6050 OffSet Calibration
imu.setXAccelOffset(-4732);
imu.setYAccelOffset(4703);
imu.setZAccelOffset(8867);
imu.setXGyroOffset(61);
imu.setYGyroOffset(-73);
imu.setZGyroOffset(35);
```

Ora, eseguire lo sketch MPU6050_Acc_Data_Acquisition.in:
Una volta eseguito lo sketch precedente, aprire il Serial Monitor:

Oppure controllare il Plotter:

Spostare il dispositivo sui tre assi. Si dovrebbe vedere la variazione sul Plotter:

Il Progetto TinyML Motion Classification

Per il nostro lab, simuleremo sollecitazioni meccaniche nel trasporto. Il nostro problema sarà classificare quattro classi di movimento:

- **Maritime** (Pallet in navi)
- **Terrestrial** (Pallet in un camion o treno)
- **Lift** (Pallet movimentati da carrello elevatore)
- **Idle** (Pallet in magazzini)

Quindi, per iniziare, dovremmo raccogliere dati. Quindi, gli accelerometri forniranno i dati sul pallet (o sul contenitore).

Dalle immagini sopra, possiamo vedere che i movimenti principalmente orizzontali dovrebbero essere associati alla “classe Terrestrial”, i movimenti verticali alla “classe Lift”, nessuna attività alla “classe Idle” e il movimento su tutti e tre gli assi alla classe Maritime.

Collegamento del dispositivo a Edge Impulse

Per la raccolta dei dati, dovremmo prima collegare il dispositivo a Edge Impulse Studio, che verrà utilizzato anche per la pre-elaborazione dei dati, l'addestramento del modello, i test e la distribuzione.

Seguire le istruzioni qui per installare Node.js e Edge Impulse CLI sul computer.

Dal momento che XIAO ESP32S3 non è più una scheda di sviluppo completamente supportata da Edge Impulse, dovremmo, ad esempio, usare CLI Data Forwarder per acquisire dati dal nostro sensore e inviarli allo Studio, come mostrato in questo diagramma:

In alternativa, si possono acquisire i dati "offline", memorizzarli su una scheda SD o inviarli al computer tramite Bluetooth o Wi-Fi. In questo video, si possono imparare modi alternativi per inviare dati a Edge Impulse Studio.

Collegare il dispositivo alla porta seriale ed eseguire il codice precedente per catturare i dati IMU (Accelerometro), "stampandoli" sulla seriale. Questo consentirà a Edge Impulse Studio di "catturarli".
Si va alla pagina Edge Impulse e si crea un progetto.

La lunghezza massima per un nome di libreria Arduino è di **63 caratteri**. Notare che Studio nominerà la libreria finale usando il nome del progetto e vi includerà "_inference". Il nome scelto inizialmente non ha funzionato quando è stato provato a distribuire la libreria Arduino perché risultava in 64 caratteri. Quindi, lo si deve cambiare eliminando la parte "anomaly detection".

Avviare CLI Data Forwarder sul terminale, immettendo (se è la prima volta) il seguente comando:

```
edge-impulse-data-forwarder --clean
```

Quindi, immettere le credenziali EI e scegli il progetto, le variabili e i nomi dei dispositivi:

Si va al progetto EI e si verifica se il dispositivo è connesso (il punto dovrebbe essere verde):

Raccolta Dati

Come discusso in precedenza, dovremmo catturare dati da tutte e quattro le classi Transportation". Immaginiamo di avere un contenitore con un accelerometro incorporato:

Ora immaginiamo che il contenitore sia su una barca, di fronte a un oceano in tempesta, su un camion, ecc.:

- **Maritime** (Pallet in navi)
  - Spostare lo XIAO in tutte le direzioni, simulando un movimento ondulatorio della barca.
- **Terrestrial** (Pallet in un camion o treno)
  - Spostare lo XIAO su una linea orizzontale.
- **Lift** (Pallet movimentati da carrello elevatore)
  - Spostare lo XIAO su una linea verticale.
- **Idle** (Pallet in magazzini)
  - Lasciare lo XIAO sul tavolo.

Di seguito è riportato un campione (dati grezzi) di 10 secondi:

Ad esempio, si possono catturare 2 minuti (dodici campioni da 10 secondi ciascuno) per le quattro classi. Utilizzando i “3 punti” dopo ciascuno dei campioni, selezionarne 2, spostandoli per il set di test (o utilizzare lo strumento automatico Train/Test Split nella scheda Danger Zone della Dashboard). Di seguito, si possono vedere i dataset dei risultati:

Pre-elaborazione dei Dati

Il tipo di dati grezzi catturato dall'accelerometro è una "serie temporale" e dovrebbe essere convertito in "dati tabulari". Possiamo effettuare questa conversione utilizzando una finestra scorrevole sui dati campione. Ad esempio, nella figura sottostante,

Possiamo vedere 10 secondi di dati dell'accelerometro catturati con un "sample rate (SR)" [frequenza di campionamento] di 50 Hz. Una finestra di 2 secondi catturerà 300 dati (3 assi x 2 secondi x 50 campioni). Faremo scorrere questa finestra ogni 200 ms, creando un set di dati più grande in cui ogni istanza ha 300 feature grezze.

> Si deve usare il miglior SR per il proprio caso, considerando il teorema di Nyquist, che afferma che un segnale periodico deve essere campionato a più del doppio della componente di frequenza più alta del segnale.

La pre-elaborazione dei dati è un'area impegnativa per l'apprendimento automatico embedded. Tuttavia, Edge Impulse aiuta a superare questo problema con la sua fase di pre-elaborazione dell'elaborazione del segnale digitale (DSP) e, più specificamente, le Spectral Feature.

Su Studio, questo dataset sarà l'input di un blocco Spectral Analysis, che è eccellente per analizzare il movimento ripetitivo, come i dati degli accelerometri. Questo blocco eseguirà un DSP (Digital Signal Processing), estraendo caratteristiche come "FFT" o "Wavelets". Nel caso più comune, FFT, le **le feature Time Domain Statistical** per asse/canale sono:

- RMS
- Skewness
- Kurtosis

E la **Frequency Domain Spectral features** per asse/canale sono:

- Spectral Power [Potenza spettrale]
- Skewness
- Kurtosis

Ad esempio, per una lunghezza FFT di 32 punti, l'output risultante del blocco di analisi spettrale sarà di 21 feature per asse (un totale di 63 feature).
Quelle 63 feature saranno il tensore di input di un classificatore di reti neurali e il modello di rilevamento delle anomalie (K-Means).

> Se ne può sapere di più esplorando il laboratorio DSP Spectral Features

Progettazione del Modello

Il nostro classificatore sarà una Dense Neural Network (DNN) che avrà 63 neuroni sul suo layer di input, due layer nascosti con 20 e 10 neuroni e un layer di output con quattro neuroni (uno per ogni classe), come mostrato qui:

Impulse Design

Un impulso prende dati grezzi, usa l'elaborazione del segnale per estrarre le caratteristiche e poi usa un blocco di apprendimento per classificare nuovi dati.
Sfruttiamo anche un secondo modello, il K-means, che può essere usato per le "Anomaly Detection" rilevamento delle anomalie. Se immaginiamo di poter avere le nostre classi note come cluster, qualsiasi campione che non potrebbe adattarsi potrebbe essere un valore anomalo, un'anomalia (ad esempio, un container che rotola fuori da una nave in mare).

Immaginiamo il nostro XIAO che rotola o si muove capovolto, su un complemento di movimento diverso da quello addestrato

Di seguito è riportato il nostro progetto Impulse finale:

Generazione di feature

A questo punto del nostro progetto, abbiamo definito il metodo di pre-elaborazione e il modello progettato. Ora è il momento di terminare il lavoro. Per prima cosa, prendiamo i dati grezzi (tipo serie temporale) e convertiamoli in dati tabulari. Andiamo alla scheda Spectral Features e selezioniamo Save Parameters:

Nel menù in alto, selezionare l'opzione Generate Features e il pulsante Generate Features. Ogni dato della finestra di 2 secondi verrà convertito in un punto dati di 63 feature.

> Feature Explorer mostrerà quei dati in 2D usando UMAP. Uniform Manifold Approximation and Projection (UMAP) è una tecnica di riduzione delle dimensioni che può essere usata per la visualizzazione in modo simile a t-SNE ma anche per la riduzione generale delle dimensioni non lineari.

La visualizzazione consente di verificare che le classi presentino un'eccellente separazione, il che indica che il classificatore dovrebbe funzionare bene.

Facoltativamente, si può analizzare l'importanza relativa di ogni feature per una classe rispetto ad altre classi.

Training

Il nostro modello ha quattro layer, come mostrato di seguito:

Come iperparametri, useremo un Learning Rate di 0,005 e il 20% di dati per la convalida per 30 epoche. Dopo l'addestramento, possiamo vedere che l'accuratezza è del 97%.

Per il rilevamento delle anomalie, dovremmo scegliere le feature suggerite che sono esattamente le più importanti nell'estrazione delle feature. Il numero di cluster sarà 32, come suggerito dallo Studio:

Test

Utilizzando il 20% dei dati lasciati indietro durante la fase di acquisizione dati, possiamo verificare come si comporterà il nostro modello con dati sconosciuti; se non al 100% (come previsto), il risultato non è stato così buono (8%), principalmente a causa della classe "terrestrial". Una volta che abbiamo quattro classi (il cui output dovrebbe aggiungere 1,0), possiamo impostare una soglia inferiore affinché una classe sia considerata valida (ad esempio, 0.4):

Ora, l'accuratezza del test salirà al 97%.

Si deve anche usare il dispositivo (che è ancora connesso allo Studio) ed eseguire una "Live Classification".

> Notare che qui si cattureranno dati reali con il dispositivo e si caricheranno sullo Studio, dove verrà presa un'inferenza usando il modello addestrato (ma il modello NON è nel dispositivo).

Distribuzione

Adesso è il momento della magia! Studio impacchetterà tutte le librerie necessarie, le funzioni di pre-elaborazione e i modelli addestrati, scaricandoli sul computer. Si deve selezionare l'opzione "Arduino Library" e, in basso, scegliere Quantized (Int8) e Build. Verrà creato un file Zip e scaricato sul computer.

Sull'IDE Arduino, si va alla scheda Sketch, si seleziona l'opzione Add.ZIP Library e si sceglie il file.zip scaricato da Studio:

Inferenza

Ora è il momento di un vero test. Faremo inferenze completamente scollegate dallo Studio. Modifichiamo uno degli esempi di codice creati quando si distribuisce la libreria Arduino. Nell'IDE Arduino, si va alla scheda File/Examples e si cerca il progetto, e negli esempi, si seleziona nano_ble_sense_accelerometer:

Ovviamente, questa non è la propria board, ma possiamo far funzionare il codice con solo poche modifiche.

Ad esempio, all'inizio del codice, c'è la libreria relativa ad Arduino Sense IMU:

```
/* Includes ------------------------------------------------------------------ */
#include <XIAO-ESP32S3-Motion-Classification_inferencing.h>
#include <Arduino_LSM9DS1.h>
```

Sostituire la parte "include" con il codice relativo all'IMU:

```
#include <XIAO-ESP32S3-Motion-Classification_inferencing.h>
#include "I2Cdev.h"
#include "MPU6050.h"
#include "Wire.h"
```

Cambiare le Constant Defines

```
/* Constant defines ---------------------------------------------------------- */
MPU6050 imu;
int16_t ax, ay, az;

#define ACC_RANGE           1 // 0: -/+2G; 1: +/-4G
#define CONVERT_G_TO_MS2    (9.81/(16384/(1.+ACC_RANGE)))
#define MAX_ACCEPTED_RANGE  (2*9.81)+(2*9.81)*ACC_RANGE
```

Nella funzione di configurazione, avviare l'IMU per impostare i valori di offset e il range:

```
// initialize device
Serial.println("Initializing I2C devices...");
Wire.begin();
imu.initialize();
delay(10);

//Set MCU 6050 OffSet Calibration
imu.setXAccelOffset(-4732);
imu.setYAccelOffset(4703);
imu.setZAccelOffset(8867);
imu.setXGyroOffset(61);
imu.setYGyroOffset(-73);
imu.setZGyroOffset(35);

imu.setFullScaleAccelRange(ACC_RANGE);
```

Nella funzione loop, i buffer buffer[ix], buffer[ix + 1] e buffer[ix + 2] riceveranno i dati a 3 assi catturati dall'accelerometro. Nel codice originale, si ha la riga:

```
IMU.readAcceleration(buffer[ix], buffer[ix + 1], buffer[ix + 2]);
```

Modificarlo con questo blocco di codice:

```
imu.getAcceleration(&ax, &ay, &az);
buffer[ix + 0] = ax;
buffer[ix + 1] = ay;
buffer[ix + 2] = az;
```

Si deve cambiare l'ordine dei due blocchi di codice seguenti. Per prima cosa, si esegue la conversione in dati grezzi in "Meters per squared second (ms^2)", seguita dal test riguardante il range di accettazione massimo (che qui è in ms^2, ma su Arduino era in Gs):

```
buffer[ix + 0] *= CONVERT_G_TO_MS2;
buffer[ix + 1] *= CONVERT_G_TO_MS2;
buffer[ix + 2] *= CONVERT_G_TO_MS2;

for (int i = 0; i < 3; i++) {
     if (fabs(buffer[ix + i]) > MAX_ACCEPTED_RANGE) {
        buffer[ix + i] = ei_get_sign(buffer[ix + i]) * MAX_ACCEPTED_RANGE;
     }
}
```

E questo è tutto! Ora si può caricare il codice sul dispositivo e procedere con le inferenze. Il codice completo è disponibile sul GitHub del progetto.

Ora si devono provare i movimenti, osservando il risultato dell'inferenza di ogni classe sulle immagini:

E, naturalmente, qualche “anomalia”, ad esempio, capovolgere lo XIAO. Il punteggio anomalo sarà superiore a 1:

Conclusione

Per quanto riguarda l’IMU, questo progetto ha utilizzato l’MPU6050 a basso costo, ma potrebbe utilizzare anche altre IMU, ad esempio l’LCM20600 (6 assi), che fa parte del Seeed Grove - IMU 9DOF (lcm20600+AK09918). Si può sfruttare questo sensore, che ha integrato un connettore Grove, che può essere utile nel caso in cui si utilizza lo XIAO con una scheda di estensione, come mostrato di seguito:

Si possono seguire le istruzioni qui per collegare l'IMU alla MCU. Notare solo che per usare l'accelerometro Grove ICM20600, è essenziale aggiornare i file **I2Cdev.cpp** e **I2Cdev.h** scaricabili dalla libreria fornita da Seeed Studio. Per farlo, si sostituiscono entrambi i file da questo link. Si trova uno sketch per testare l'IMU sul progetto GitHub: accelerometer_test.ino.

> Nel repository GitHub del progetto, si trova l'ultima versione di tutto il codice e altri documenti: XIAO-ESP32S3 - IMU.

Risorse

- Codici XIAO ESP32S3
- Edge Impulse Spectral Features Block Colab Notebook
- Progetto Edge Impulse

Raspberry Pi

Questi lab offrono un'esperienza pratica inestimabile con i sistemi di apprendimento automatico, sfruttando la versatilità e l'accessibilità della piattaforma Raspberry Pi. A differenza del lavoro con modelli su larga scala che richiedono ampie risorse cloud, questi esercizi consentono di interagire direttamente con hardware e software in un ambiente di edge computing compatto ma potente. Si otterranno informazioni pratiche sull'implementazione dell'intelligenza artificiale in edge utilizzando le capacità di Raspberry Pi, dall'efficiente Pi Zero ai modelli più robusti Pi 4 o Pi 5. Questo approccio fornisce una comprensione tangibile delle sfide e delle opportunità nell'implementazione di soluzioni di apprendimento automatico in contesti con risorse limitate. Mentre lavoriamo su scala più piccola, i principi e le tecniche che si impareranno sono fondamentalmente simili a quelli utilizzati nei sistemi più grandi. La capacità di Raspberry Pi di eseguire un intero sistema operativo e le sue ampie capacità GPIO consentono un'esperienza di apprendimento ricca che colma il divario tra conoscenza teorica e applicazione nel mondo reale. Attraverso questi lab, si comprenderanno le complessità di EdgeML e si svilupperanno competenze applicabili a un'ampia gamma di scenari di distribuzione dell'IA.

Raspberry Pi Zero 2-W e Raspberry Pi 5 con Telecamera

21.7 Prerequisiti

- **Raspberry Pi**: Si deve avere almeno una delle seguenti schede: Raspberry Pi Zero 2W, Raspberry Pi 4 o 5 per Vision Labs e Raspberry 5 per GenAi Lab.
- **Adattatore di Alimentazione**: Per alimentare le schede.
  - Raspberry Pi Zero 2-W: 2,5 W con un adattatore Micro-USB
  - Raspberry Pi 4 o 5: 3,5 W con un adattatore USB-C
- **Rete**: Con accesso a Internet per scaricare il software necessario e controllare le schede da remoto.
- **Scheda SD (minimo 32 GB) e un Adattatore per Schede SD**: Per il sistema operativo Raspberry Pi.

21.8 Setup

- Configurazione di Raspberry Pi

21.9 Esercizi

Modalità	Attività	Descrizione	Link
Visione	Classificazione delle Immagini	Imparare a classificare le immagini	Link
Visione	Rilevamento degli Oggetti	Implementare il rilevamento degli oggetti	Link
GenAI	Small Language Models	Deploy SLMs at the Edge	Link

Setup

DALL·E prompt - Un ambiente di laboratorio di elettronica ispirato agli anni '50, con uno stile cartoon. Il laboratorio dovrebbe avere apparecchiature vintage, grandi oscilloscopi, vecchie radio a valvole e grandi computer squadrati. La scheda Raspberry Pi 5 è esposta in modo prominente, mostrata accuratamente nelle sue dimensioni reali, simile a una carta di credito, su un banco da lavoro. La scheda Pi è circondata da classici strumenti da laboratorio come un saldatore, resistenze e fili. La scena generale dovrebbe essere vivace, con colori esagerati e dettagli giocosi caratteristici di un cartone animato. Non dovrebbero essere inclusi loghi o testo.

Questo capitolo guiderà attraverso la configurazione dei modelli Raspberry Pi Zero 2 W (*Raspi-Zero*) e Raspberry Pi 5 (*Raspi-5*). Tratteremo la configurazione hardware, l'installazione del sistema operativo, la configurazione iniziale e i test.

> Le istruzioni generali per *Raspi-5* si applicano anche alle versioni più vecchie di Raspberry Pi, come Raspi-3 e Raspi-4.

Introduzione

Il Raspberry Pi è un potente e versatile computer a scheda singola che è diventato uno strumento essenziale per gli ingegneri di varie discipline. Sviluppati dalla Raspberry Pi Foundation, questi dispositivi compatti offrono una combinazione unica di convenienza,

potenza di calcolo e ampie capacità GPIO (General Purpose Input/Output), rendendoli ideali per la prototipazione, lo sviluppo di sistemi embedded e progetti di ingegneria avanzata.

Caratteristiche Principali

1. **Potenza di Calcolo**: Nonostante le dimensioni ridotte, Raspberry Pi offre notevoli capacità di elaborazione, con gli ultimi modelli dotati di processori ARM multi-core e fino a 8 GB di RAM.
2. **Interfaccia GPIO**: L'header GPIO a 40 pin consente l'interazione diretta con sensori, attuatori e altri componenti elettronici, facilitando i progetti di integrazione hardware-software.
3. **Ampia Connettività**: Wi-Fi, Bluetooth, Ethernet e più porte USB integrate consentono diversi progetti di comunicazione e networking.
4. **Accesso Hardware di Basso Livello**: Raspberry Pi fornisce accesso a interfacce come I2C, SPI e UART, consentendo un controllo dettagliato e la comunicazione con dispositivi esterni.
5. **Capacità in Tempo Reale**: Con una configurazione appropriata, Raspberry Pi può essere utilizzato per applicazioni soft in tempo reale, rendendolo adatto per sistemi di controllo e attività di elaborazione del segnale.
6. **Efficienza Energetica**: Il basso consumo energetico consente progetti alimentati a batteria e a basso consumo energetico, in particolare in modelli come Pi Zero.

Modelli Raspberry Pi (trattati in questo libro)

1. **Raspberry Pi Zero 2 W** (*Raspi-Zero*):
   - Ideale per: sistemi embedded compatti
   - Specifiche principali: CPU single-core da 1 GHz (ARM Cortex-A53), 512 MB di RAM, consumo energetico minimo
2. **Raspberry Pi 5** (*Raspi-5*):
   - Ideale per: applicazioni più esigenti come edge computing, computer vision e applicazioni edgeAI, inclusi LLM.
   - Specifiche principali: CPU quad-core da 2,4 GHz (ARM Cortex A-76), fino a 8 GB di RAM, interfaccia PCIe per espansioni

Applicazioni di Ingegneria

1. **Progettazione di Sistemi Embedded**: Sviluppo e prototipi di sistemi embedded per applicazioni reali.
2. **IoT e Dispositivi in Rete**: Creazione di dispositivi interconnessi ed esplorazione di protocolli come MQTT, CoAP e HTTP/HTTPS.
3. **Sistemi di Controllo**: Implementazione di loop di controllo feedback, controller PID e interfaccia per attuatori.
4. **Visione Artificiale e IA**: Utilizzo di librerie come OpenCV e TensorFlow Lite per l'elaborazione delle immagini e l'apprendimento automatico in edge.
5. **Acquisizione e Analisi dei Dati**: Raccolta dati dei sensori, analisi in tempo reale e creazione di sistemi di registrazione dei dati.
6. **Robotica**: Creazione di controller per robot, algoritmi di pianificazione del movimento e interfaccie con driver di motori.
7. **Elaborazione del Segnale**: Analisi del segnale in tempo reale, filtraggio e applicazioni DSP.
8. **Sicurezza di Rete**: Imposta VPN, firewall ed esplora i test di penetrazione della rete.

Questo tutorial guiderà nell'impostazione dei modelli Raspberry Pi più comuni, consentendo di iniziare rapidamente il progetto di apprendimento automatico. Tratteremo la

configurazione hardware, l'installazione del sistema operativo e la configurazione iniziale, concentrandoci sulla preparazione del Pi per le applicazioni di apprendimento automatico.

Panoramica Hardware

Raspberry Pi Zero 2W

- **Processore**: CPU Arm Cortex-A53 quad-core a 64 bit da 1 GHz
- **RAM**: SDRAM da 512 MB
- **Wireless**: LAN wireless 802.11 b/g/n da 2,4 GHz, Bluetooth 4.2, BLE
- **Porte**: Mini HDMI, micro USB OTG, connettore per fotocamera CSI-2
- **Alimentazione**: 5 V tramite porta micro USB

Raspberry Pi 5

- **Processore**:
  - Pi 5: CPU Arm Cortex-A76 quad-core a 64 bit @ 2,4 GHz
  - Pi 4: SoC Cortex-A72 quad-core (ARM v8) a 64 bit @ 1,5 GHz
- **RAM**: opzioni da 2 GB, 4 GB o 8 GB (8 GB consigliati per attività di IA)
- **Wireless**: Wireless 802.11ac dual-band, Bluetooth 5.0

- **Porte**: 2 porte micro HDMI, 2 porte USB 3.0, 2 porte USB 2.0, porta fotocamera CSI, porta display DSI
- **Alimentazione**: 5 V CC tramite connettore USB-C (3 A)

Nei laboratori, useremo nomi diversi per riferirci al Raspberry: `Raspi`, `Raspi-5`, `Raspi-Zero`, ecc. Di solito, si usa `Raspi` quando le istruzioni o i commenti si applicano a tutti i modelli.

Installazione del Sistema Operativo

Il sistema operativo (SO)

Un sistema operativo (SO) è un software fondamentale che gestisce le risorse hardware e software del computer, fornendo servizi standard per i programmi per computer. È il software principale che gira su un computer, fungendo da intermediario tra hardware e software applicativo. Il SO gestisce la memoria, i processi, i driver dei dispositivi, i file e i protocolli di sicurezza del computer.

1. **Funzioni chiave**:
    - Gestione dei processi: Assegnazione del tempo della CPU a diversi programmi
    - Gestione della memoria: Assegnazione e liberazione della memoria in base alle necessità
    - Gestione del file system: Organizzazione e monitoraggio di file e directory
    - Gestione dei dispositivi: Comunicazione con dispositivi hardware collegati
    - Interfaccia utente: Fornitura di un modo per gli utenti di interagire con il computer
2. **Componenti**:
    - Kernel: Il nucleo del sistema operativo che gestisce le risorse hardware
    - Shell: L'interfaccia utente per interagire con il sistema operativo
    - File system: Organizzazione e gestione dell'archiviazione dei dati
    - Device driver: Software che consente al sistema operativo di comunicare con l'hardware

Il Raspberry Pi esegue una versione specializzata di Linux progettata per sistemi embedded. Questo sistema operativo, in genere una variante di Debian chiamata Raspberry Pi OS (in precedenza Raspbian), è ottimizzato per l'architettura basata su ARM del Pi e per le risorse limitate.

L'ultima versione di Raspberry Pi OS è basata su Debian Bookworm.

Caratteristiche principali:

1. Leggero: Progettato per funzionare in modo efficiente sull'hardware del Pi.
2. Versatile: Supporta un'ampia gamma di applicazioni e linguaggi di programmazione.
3. Open source: Consente la personalizzazione e i miglioramenti guidati dalla comunità.
4. Supporto GPIO: Consente l'interazione con sensori e altri hardware tramite i pin del Pi.
5. Aggiornamenti regolari: Costantemente migliorato per prestazioni e sicurezza.

Embedded Linux sul Raspberry Pi fornisce un sistema operativo completo in un pacchetto compatto, rendendolo ideale per progetti che vanno da semplici dispositivi IoT ad applicazioni di apprendimento automatico edge più complesse. La sua compatibilità con gli strumenti e le librerie Linux standard lo rende una potente piattaforma per lo sviluppo e la sperimentazione.

Installazione

Per usare Raspberry Pi, avremo bisogno di un sistema operativo. Di default, Raspberry Pi verifica la presenza di un sistema operativo su qualsiasi scheda SD inserita nello slot, quindi dovremmo installare un sistema operativo usando Raspberry Pi Imager.
Raspberry Pi Imager è uno strumento per scaricare e scrivere immagini su *macOS*, *Windows* e *Linux*. Include molte immagini di sistemi operativi popolari per Raspberry Pi. Useremo Imager anche per preconfigurare credenziali e impostazioni di accesso remoto.
Seguire i passaggi per installare il sistema operativo nel Raspi.

1. Scaricare e installare Raspberry Pi Imager sul computer.
2. Inserire una scheda microSD nel computer (si consiglia una scheda SD da 32 GB).
3. Aprire Raspberry Pi Imager e selezionare il modello di Raspberry Pi.
4. Scegliere il sistema operativo appropriato:
    - **For Raspi-Zero**: Ad esempio, si può selezionare: `Raspberry Pi OS Lite (64-bit)`.

img

Grazie alla ridotta SDRAM (512 MB), il sistema operativo consigliato per Raspi-Zero è la versione a 32 bit. Tuttavia, per eseguire alcuni modelli di apprendimento automatico, come YOLOv8 di Ultralitics, dovremmo usare la versione a 64 bit. Sebbene Raspi-Zero possa eseguire un *desktop*, sceglieremo la versione LITE (senza Desktop) per ridurre la RAM necessaria per il normale funzionamento.

    - Per **Raspi-5**: possiamo selezionare la versione completa a 64 bit, che include un desktop: `Raspberry Pi OS (64-bit)`

5. Selezionare la scheda microSD come dispositivo di archiviazione.
6. Cliccare su `Next` e poi sull'icona dell'`ingranaggio` per accedere alle opzioni avanzate.
7. Impostare *hostname*, *nome utente* e *password* di Raspi, configurare il *WiFi* e *abilitare SSH* (molto importante!)

8. Scrivere l'immagine sulla scheda microSD.

Negli esempi qui, useremo nomi host diversi a seconda del dispositivo utilizzato: raspi, raspi-5, raspi-Zero, ecc. Sarebbe utile se lo si sostituisse con quello in uso.

Configurazione Iniziale

1. Inserire la scheda microSD nel Raspberry Pi.
2. Collegare l'alimentazione per avviare il Raspberry Pi.

3. Attendere il completamento del processo di boot iniziale (potrebbero volerci alcuni minuti).

I comandi Linux più comuni da usare con Raspi si trovano qui o qui.

Accesso Remoto

Accesso SSH

Il modo più semplice per interagire con Raspi-Zero è tramite SSH ("Headless"). Si può usare un terminale (MAC/Linux), PuTTy (Windows) o qualsiasi altro.

1. Trovare l'indirizzo IP del Raspberry Pi (ad esempio, controllando il router).
2. Sul computer, aprire un terminale e connettersi tramite SSH:

```
ssh username@[raspberry_pi_ip_address]
```

In alternativa, se non si ha l'indirizzo IP, si può provare quanto segue:

```
ssh username@hostname.local
```

per esempio, `ssh mjrovai@rpi-5.local` , `ssh mjrovai@raspi.local` , ecc.

img

Quando si vede il prompt:

```
mjrovai@rpi-5:~ $
```

Significa che si sta interagendo da remoto con il Raspi. È una buona norma aggiornare/migliorare il sistema regolarmente. Per questo, si deve eseguire:

```
sudo apt-get update
sudo apt upgrade
```

Si deve confermare l'indirizzo IP Raspi. Sul terminale, si può usare:

```
hostname -I
```

Per spegnere il Raspi tramite terminale:

Per spegnere il Raspberry Pi, ci sono idee migliori che staccare semplicemente il cavo di alimentazione. Questo perché il Raspi potrebbe ancora scrivere dati sulla scheda SD, nel qual caso il semplice spegnimento potrebbe causare la perdita di dati o, peggio ancora, una scheda SD danneggiata.
Per uno spegnimento di sicurezza, usare la riga di comando:

```
sudo shutdown -h now
```

> Per evitare possibili perdite di dati e danneggiamenti della scheda SD, prima di rimuovere l'alimentazione, si deve attendere qualche secondo dopo lo spegnimento affinché il LED del Raspberry Pi smetta di lampeggiare e si spenga. Una volta che il LED si spegne, è sicuro spegnere.

Trasferire file tra il Raspi e un computer

Il trasferimento di file tra il Raspi e il nostro computer principale può essere effettuato tramite una chiavetta USB, direttamente sul terminale (con scp) o un programma FTP sulla rete.

Utilizzo del Protocollo Secure Copy Protocol (`scp`):

Copiare i file sul Raspberry Pi

Creiamo un file di testo sul nostro computer, ad esempio, `test.txt`.

> Si può usare qualsiasi editor di testo. Nello stesso terminale, un'opzione è `nano`.

Per copiare il file `test.txt` dal personal computer alla cartella home di un utente sul Raspberry Pi, si esegue il seguente comando dalla directory contenente `test.txt`, sostituendo il segnaposto `<username>` con il nome utente che si usa per accedere al Raspberry Pi e il segnaposto `<pi_ip_address>` con l'indirizzo IP del Raspberry Pi:

```
$ scp test.txt <username>@<pi_ip_address>:~/
```

> Notare che `~/` significa che sposteremo il file nella ROOT del Raspi. Si può scegliere qualsiasi cartella nel Raspi. Ma si deve creare la cartella prima di eseguire `scp`, poiché `scp` non lo fa automaticamente.

Ad esempio, trasferiamo il file `test.txt` nella ROOT dell'Raspi-zero, che ha un IP di `192.168.4.210`:

```
scp test.txt mjrovai@192.168.4.210:~/
```

Abbiamo usato un profilo diverso per differenziare i terminali. L'azione di cui sopra avviene **sul computer**. Ora, andiamo sul Raspi (utilizzando SSH) e controlliamo se il file è lì:

Copiare i file dal Raspberry Pi

Per copiare un file chiamato `test.txt` dalla directory home di un utente su un Raspberry Pi alla directory corrente su un altro computer, si esegue il seguente comando **sul computer host**:

```
$ scp <username>@<pi_ip_address>:myfile.txt .
```

Per esempio:

Sul Raspi, creiamo una copia del file con un altro nome:

```
cp test.txt test_2.txt
```

E sul computer host (in questo caso, un Mac)

```
scp mjrovai@192.168.4.210:test_2.txt .
```

Trasferimento di file tramite FTP

È anche possibile trasferire file tramite FTP, come FileZilla FTP Client. Seguire le istruzioni, installare il programma per il proprio sistema operativo desktop e usare l'indirizzo IP Raspi come `Host`. Per esempio:

```
sftp://192.168.4.210
```

e inserire `nome utente e password` di Raspi . Premendo `Quickconnect` si apriranno due finestre, una per il desktop del computer host (destra) e un'altra per il Raspi (sinistra).

Aumentare la memoria SWAP

Utilizzando `htop`, un visualizzatore di processi interattivo multipiattaforma, si possono facilmente monitorare, in tempo reale, le risorse in esecuzione sul Raspi, come l'elenco dei processi, le CPU in esecuzione e la memoria utilizzata. Per lanciare `hop`, si immette il vomando sul terminale:

```
htop
```

Per quanto riguarda la memoria, tra i dispositivi della famiglia Raspberry Pi, il Raspi-Zero ha la quantità più piccola di SRAM (500 MB), rispetto a una selezione da 2 GB a 8 GB sui Raspi 4 o 5. Per qualsiasi Raspi, è possibile aumentare la memoria disponibile per il sistema con "Swap". La memoria di swap, nota anche come spazio di swap, è una tecnica utilizzata nei sistemi operativi dei computer per archiviare temporaneamente i dati dalla RAM (Random Access Memory) sulla scheda SD quando la RAM fisica è completamente utilizzata. Ciò consente al sistema operativo (SO) di continuare a funzionare anche quando la RAM è piena, il che può prevenire crash o rallentamenti del sistema.
La memoria di swap avvantaggia i dispositivi con RAM limitata, come il Raspi-Zero. Aumentare lo swap può aiutare a eseguire applicazioni o processi più impegnativi, ma è essenziale bilanciare questo con il potenziale impatto sulle prestazioni dell'accesso frequente al disco.
Per default, la memoria SWAP (Swp) di Rapi-Zero è di soli 100 MB, il che è molto poco per l'esecuzione di alcune applicazioni di apprendimento automatico più complesse ed esigenti (ad esempio, YOLO). Aumentiamola a 2 MB:
Per prima cosa, si disattiva lo swap-file:

```
sudo dphys-swapfile swapoff
```

Poi, si deve aprire e modificare il file `/etc/dphys-swapfile`. Per farlo, useremo nano:

```
sudo nano /etc/dphys-swapfile
```

Cercare la variabile **CONF_SWAPSIZE** (il valore predefinito è 200) e aggiornarla a **2000**:

```
CONF_SWAPSIZE=2000
```

E salvare il file.
Quindi, riattivare lo swapfile e riavviare Raspi-zero:

```
sudo dphys-swapfile setup
sudo dphys-swapfile swapon
sudo reboot
```

Quando il dispositivo viene riavviato (so deve entrare di nuovo con SSH), ci si accorgerà che il valore massimo della memoria di swap mostrato in alto è ora qualcosa come 2 GB (nel nostro caso, 1.95 GB).

> Per mantenere in esecuzione *htop*, si deve aprire un'altra finestra del terminale per interagire continuamente con il Raspi.

Installazione di una Fotocamera

Il Raspi è un dispositivo eccellente per applicazioni di visione artificiale; è necessaria una fotocamera. Possiamo installare una webcam USB standard sulla porta micro-USB utilizzando un adattatore USB OTG (Raspi-Zero e Raspi-5) oppure un modulo telecamera collegato alla porta Raspi CSI (Camera Serial Interface).

> Le webcam USB hanno generalmente una qualità inferiore rispetto ai moduli fotocamera che si collegano alla porta CSI. Inoltre, non possono essere controllate utilizzando i comandi `raspistill` e `raspivid` nel terminale o il pacchetto di registrazione `picamera` in Python. Tuttavia, potrebbero esserci dei motivi per collegare una telecamera USB al Raspberry Pi, ad esempio perché è molto più facile configurare più telecamere con un singolo Raspberry Pi, cavi lunghi o semplicemente perché si ha una telecamera del genere a portata di mano.

Installazione di una Webcam USB

1. Spegnere il Raspi:

```
sudo shutdown -h no
```

2. Collegare la webcam USB (modulo telecamera USB 30fps,1280x720) al Raspi (in questo esempio, si usa il Raspi-Zero, ma le istruzioni funzionano per tutti i Raspi).

3. Riaccendere ed eseguire SSH
4. Per verificare che la fotocamera USB è riconosciuta, eseguire:

```
lsusb
```

Si dovrebbe vedere la fotocamera elencata nell'output.

5. Per scattare una foto di prova con la fotocamera USB, si usa:

```
fswebcam test_image.jpg
```

Questo salverà un'immagine denominata "test_image.jpg" nella directory corrente.

6. Dato che stiamo usando SSH per connetterci al Rapsi, dobbiamo trasferire l'immagine al computer principale in modo da poterla visualizzare. Possiamo usare FileZilla o SCP per questo:

Si apre un terminale **sul computer host** e si esegue:

```
scp mjrovai@raspi-zero.local:~/test_image.jpg .
```

Sostituire “mjrovai” col nome utente e “raspi-zero” con il nome host di Pi.

7. Se la qualità dell’immagine non è soddisfacente, si possono regolare varie impostazioni; ad esempio, definire una risoluzione adatta a YOLO (640x640):

```
fswebcam -r 640x640 --no-banner test_image_yolo.jpg
```

Questo cattura un’immagine ad alta risoluzione senza il banner di default.

Si può anche usare una normale webcam USB:

Verificandola con `lsusb`

Streaming Video

Per lo streaming video (che richiede più risorse), possiamo installare e usare mjpg-streamer: Per prima cosa, si installa Git:

```
sudo apt install git
```

Ora, dovremmo installare le dipendenze necessarie per mjpg-streamer, clonare il repository e procedere con l'installazione:

```
sudo apt install cmake libjpeg62-turbo-dev
git clone https://github.com/jacksonliam/mjpg-streamer.git
cd mjpg-streamer/mjpg-streamer-experimental
make
sudo make install
```

Quindi avviare lo streaming con:

```
mjpg_streamer -i "input_uvc.so" -o "output_http.so -w ./www"
```

Possiamo poi accedere allo streaming aprendo un browser Web e andando su: `http://<your_pi_ip_address>:8080`. In questo caso: http://192.168.4.210:8080

Dovremmo vedere una pagina web con opzioni per visualizzare lo stream. Cliccare suk che dice "Stream" o provare ad accedere a:

```
http://<raspberry_pi_ip_address>:8080/?action=stream
```

Installazione di un Modulo Fotocamera sulla porta CSI

Ora ci sono diversi moduli fotocamera Raspberry Pi. Il modello originale da 5 megapixel è stato rilasciato nel 2013, seguito da un 8-megapixel Camera Module 2 che è stato rilasciato nel 2016. L'ultimo modello di fotocamera è il 12-megapixel Camera Module 3, rilasciato nel 2023.

La fotocamera originale da 5 MP (**Arducam OV5647**) non è più disponibile da Raspberry Pi, ma può essere trovata da diversi fornitori alternativi. Di seguito è riportato un esempio di tale fotocamera su un Raspi-Zero.

Ecco un altro esempio di un modulo fotocamera v2, dotato di un sensore **Sony IMX219** da 8 megapixel:

Qualsiasi modulo fotocamera funzionerà sul Raspberry Pi, ma per questo è necessario aggiornare il file `configuration.txt`:

```
sudo nano /boot/firmware/config.txt
```

In fondo al file, ad esempio, per usare la fotocamera Arducam OV5647 da 5 MP, si deve aggiungere la riga:

```
dtoverlay=ov5647,cam0
```

Oppure per il modulo v2, che ha la fotocamera Sony IMX219 da 8 MP:

```
dtoverlay=imx219,cam0
```

Salvare il file (CTRL+O [INVIO] CRTL+X) e riavviare il Raspi:

```
Sudo reboot
```

Dopo l'avvio, si può vedere se la fotocamera è elencata:

```
libcamera-hello --list-cameras
```

libcamera è una libreria software open source che supporta i sistemi di telecamere direttamente dal sistema operativo Linux sui processori Arm. Riduce al minimo il codice proprietario in esecuzione sulla GPU Broadcom.

Catturiamo un'immagine jpeg con una risoluzione di 640 x 480 per il test e salviamola in un file denominato `test_cli_camera.jpg`

```
rpicam-jpeg --output test_cli_camera.jpg --width 640 --height 480
```

se vogliamo vedere il file salvato, dovremmo usare `ls -f`, che elenca tutto il contenuto della directory corrente in formato lungo. Come prima, possiamo usare scp per visualizzare l'immagine:

Esecuzione del desktop Raspi da remoto

Sebbene abbiamo interagito principalmente con il Raspberry Pi utilizzando comandi del terminale tramite SSH, possiamo accedere all'intero ambiente desktop grafico da remoto se abbiamo installato il sistema operativo Raspberry Pi completo (ad esempio, `Raspberry Pi OS (64-bit)`. Questo può essere particolarmente utile per le attività che traggono vantaggio da un'interfaccia visuale. Per abilitare questa funzionalità, dobbiamo configurare un server VNC (Virtual Network Computing) sul Raspberry Pi. Ecco come fare:

1. Abilitare il server VNC:
    - Connettersi al Raspberry Pi tramite SSH.
    - Esegui lo strumento di configurazione Raspberry Pi immettendo: `bash sudo raspi-config`
    - Andare su `Interface Options` utilizzando i tasti freccia.

    - Selezionare `VNC` e `Yes` per abilitare server VNC.

    - Uscire dallo strumento di configurazione, salvando le modifiche quando richiesto.

2. Installare un VNC Viewer sul computer:
   - Scaricare e installare un'applicazione di visualizzazione VNC sul computer principale. Le opzioni più diffuse includono RealVNC Viewer, TightVNC o VNC Viewer di RealVNC. Installeremo VNC Viewer di RealVNC.
3. Una volta installato, si deve confermare l'indirizzo IP Raspi. Ad esempio, sul terminale, si può usare:

```
hostname -I
```

4. Connettersi al Raspberry Pi:
   - Aprire l'applicazione di visualizzazione VNC.

   - Inserire l'indirizzo IP e il nome host del Raspberry Pi.
   - Quando richiesto, inserire il nome utente e la password del Raspberry Pi.

5. Il Raspberry Pi 5 Desktop dovrebbe apparire sul monitor del computer.

6. Regolare le Impostazioni dello Schermo (se necessario):
    - Una volta connessi, regolare la risoluzione dello schermo per una visualizzazione ottimale. Questo può essere fatto tramite le impostazioni desktop del Raspberry Pi o modificando il file config.txt.
    - Facciamolo tramite le impostazioni desktop. Raggiungere il menù (l'icona Raspberry nell'angolo in alto a sinistra) e selezionare la migliore definizione dello schermo per il proprio monitor:

Aggiornamento e Installazione del Software

1. Aggiorna il sistema:

```
sudo apt update && sudo apt upgrade -y
```

2. Installare il software essenziale:

```
sudo apt install python3-pip -y
```

3. Abilita pip per i progetti Python:

```
sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED
```

Considerazioni Specifiche del Modello

Raspberry Pi Zero (Raspi-Zero)

- Potenza di elaborazione limitata, ideale per progetti leggeri
- Per risparmiare risorse è meglio utilizzare una configurazione headless (SSH).
- Valutare l'aumento dello spazio di swap per attività che richiedono molta memoria.
- Può essere utilizzato per Image Classification e Object Detection Labs ma non per LLM (SLM).

Raspberry Pi 4 o 5 (Raspi-4 o Raspi-5)

- Adatto per progetti più impegnativi, tra cui IA e apprendimento automatico.
- Può eseguire l'intero ambiente desktop senza problemi.
- Raspi-4 può essere utilizzato per Image Classification e Object Detection Labs ma non funzionerà bene con LLM (SLM).
- Per Raspi-5, prendere in considerazione l'utilizzo di un dissipatore attivo per la gestione della temperatura durante attività intensive, come nel laboratorio LLM (SLM).

Ricordarsi di adattare i requisiti del progetto in base allo specifico modello di Raspberry Pi in uso. Raspi-Zero è ottimo per progetti a basso consumo e con vincoli di spazio, mentre i modelli Raspi-4 o 5 sono più adatti per attività più intensive dal punto di vista computazionale.

Classificazione delle Immagini

DALL·E prompt - Un'immagine di copertina per un capitolo "Classificazione delle immagini" in un tutorial Raspberry Pi, progettata nello stesso stile vintage da laboratorio di elettronica anni '50 delle copertine precedenti. La scena dovrebbe presentare un Raspberry Pi collegato a un modulo fotocamera, con la fotocamera che cattura una foto del piccolo robot blu fornito dall'utente. Il robot dovrebbe essere posizionato su un banco da lavoro, circondato da classici strumenti da laboratorio come saldatori, resistenze e fili. Lo sfondo del laboratorio dovrebbe includere apparecchiature d'epoca come oscilloscopi e radio a valvole, mantenendo il tocco dettagliato e nostalgico dell'epoca. Non dovrebbero essere inclusi testo o loghi.

Introduzione

La classificazione delle immagini è un'attività fondamentale nella visione artificiale che comporta la categorizzazione di un'immagine in una delle diverse classi predefinite. È una pietra angolare dell'intelligenza artificiale, che consente alle macchine di interpretare e comprendere le informazioni visive in un modo che imita la percezione umana.
La classificazione delle immagini si riferisce all'assegnazione di un'etichetta o di una categoria a un'intera immagine in base al suo contenuto visivo. Questa attività è fondamentale nella visione artificiale e ha numerose applicazioni in vari settori. L'importanza della classificazione delle immagini risiede nella sua capacità di automatizzare le attività di comprensione visiva che altrimenti richiederebbero l'intervento umano.

Applicazioni in Scenari del Mondo Reale

La classificazione delle immagini ha trovato la sua strada in numerose applicazioni del mondo reale, rivoluzionando vari settori:

- Sanità: Assistenza nell'analisi delle immagini mediche, come l'identificazione di anomalie nelle radiografie o nelle risonanze magnetiche.
- Agricoltura: Monitoraggio della salute delle colture e rilevamento delle malattie delle piante tramite immagini aeree.
- Automotive: Abilitazione di sistemi avanzati di assistenza alla guida e veicoli autonomi per riconoscere segnali stradali, pedoni e altri veicoli.
- Vendita al Dettaglio: Potenziamento delle capacità di ricerca visiva e sistemi di gestione automatizzata dell'inventario.
- Sicurezza e Sorveglianza: Potenziamento dei sistemi di rilevamento delle minacce e riconoscimento facciale.
- Monitoraggio Ambientale: Analisi delle immagini satellitari per studi su deforestazione, pianificazione urbana e cambiamenti climatici.

Vantaggi dell'Esecuzione della Classificazione su Dispositivi Edge come Raspberry Pi

L'implementazione della classificazione delle immagini su dispositivi edge come Raspberry Pi offre diversi vantaggi interessanti:

1. Bassa latenza: L'elaborazione delle immagini in locale elimina la necessità di inviare dati ai server cloud, riducendo significativamente i tempi di risposta.
2. Funzionalità Offline: La classificazione può essere eseguita senza una connessione Internet, rendendola adatta ad ambienti remoti o con problemi di connettività.
3. Privacy e Sicurezza: I dati sensibili delle immagini rimangono sul dispositivo locale, affrontando i problemi di privacy dei dati e i requisiti di conformità.
4. Efficacia in Termini di Costi: Elimina la necessità di costose risorse di cloud computing, in particolare per attività di classificazione continue o ad alto volume.
5. Scalabilità: Consente architetture di elaborazione distribuite in cui più dispositivi possono funzionare in modo indipendente o in una rete.
6. Efficienza Energetica: I modelli ottimizzati su hardware dedicato possono essere più efficienti dal punto di vista energetico rispetto alle soluzioni basate su cloud, il che è fondamentale per applicazioni alimentate a batteria o remote.
7. Personalizzazione: L'implementazione di modelli specializzati o aggiornati di frequente, su misura per casi d'uso specifici, è più gestibile.

Possiamo creare soluzioni di visione artificiale più reattive, sicure ed efficienti sfruttando la potenza di dispositivi edge come Raspberry Pi per la classificazione delle immagini. Questo approccio apre nuove possibilità per integrare l'elaborazione visiva intelligente in varie applicazioni e ambienti.

Nelle sezioni seguenti, esploreremo come implementare e ottimizzare la classificazione delle immagini su Raspberry Pi, sfruttando questi vantaggi per creare sistemi di visione artificiale potenti ed efficienti.

Impostazione dell'Ambiente

Aggiornamento di Raspberry Pi

Innanzitutto, assicurarsi che il Raspberry Pi sia aggiornato:

```
sudo apt update
sudo apt upgrade -y
```

Installazione delle Librerie Richieste

Installare le librerie necessarie per l'elaborazione delle immagini e l'apprendimento automatico:

```
sudo apt install python3-pip
sudo rm /usr/lib/python3.11/EXTERNALLY-MANAGED
pip3 install --upgrade pip
```

Impostazione di un Ambiente Virtuale (Facoltativo ma Consigliato)

Creare un ambiente virtuale per gestire le dipendenze:

```
python3 -m venv ~/tflite
source ~/tflite/bin/activate
```

Installazione di TensorFlow Lite

Siamo interessati a eseguire **inferenza**, ovvero l'esecuzione di un modello TensorFlow Lite su un dispositivo per effettuare previsioni basate sui dati di input. Per eseguire un'inferenza con un modello TensorFlow Lite, dobbiamo eseguirla tramite un **interprete**. L'interprete TensorFlow Lite è progettato per essere snello e veloce. L'interprete utilizza un ordinamento grafico statico e un allocatore di memoria personalizzato (meno dinamico) per garantire un carico, inizializzazione ed esecuzione latenza minimi.
Utilizzeremo il runtime TensorFlow Lite per Raspberry Pi, una libreria semplificata per l'esecuzione di modelli di apprendimento automatico su dispositivi mobili e embedded, senza includere tutti i pacchetti TensorFlow.

```
pip install tflite_runtime --no-deps
```

La wheel installata: `tflite_runtime-2.14.0-cp311-cp311-manylinux_2_34_aarch64.whl`

Installazione di Librerie Python Aggiuntive

Installare le librerie Python richieste per l'uso con Image Classification:
Se è installata un'altra versione di Numpy, disinstallarla prima.

```
pip3 uninstall numpy
```

Installare la `versione 1.23.2`, che è compatibile con tflite_runtime.

```
pip3 install numpy==1.23.2
pip3 install Pillow matplotlib
```

Creazione di una directory di lavoro:

Se si lavora su Raspi-Zero con il sistema operativo minimo (No Desktop), non si ha un albero di directory user-pre-defined (lo si può verificare con `ls`. Quindi, creiamone uno:

```
mkdir Documents
cd Documents/
mkdir TFLITE
cd TFLITE/
mkdir IMG_CLASS
cd IMG_CLASS
mkdir models
cd models
```

Su Raspi-5, /Documents dovrebbe esserci.

Ottenere un Modello di Classificazione delle Immagini Pre-addestrato:

Un modello pre-addestrato appropriato è fondamentale per una classificazione delle immagini di successo su dispositivi con risorse limitate come Raspberry Pi. **MobileNet** è progettato per applicazioni di visione mobile e embedded con un buon equilibrio tra accuratezza e velocità. Versioni: MobileNetV1, MobileNetV2, MobileNetV3. Scarichiamo la V2:

```
wget https://storage.googleapis.com/download.tensorflow.org/models/
tflite_11_05_08/mobilenet_v2_1.0_224_quant.tgz

tar xzf mobilenet_v2_1.0_224_quant.tgz
```

Prelevarne le etichette:

```
wget https://github.com/Mjrovai/EdgeML-with-Raspberry-Pi/blob/main/IMG_CLASS/models/labels.txt
```

Alla fine, si dovrebbero avere i modelli nella sua directory:

```
marcelo_rovai — mjrovai@raspi-zero: ~/Documents/TFLITE/IMG_CLASS/models — ssh mjrovai@192...
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS $ cd models
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS/models $ ls
labels.txt
mobilenet_v2_1.0_224_quant.ckpt.data-00000-of-00001
mobilenet_v2_1.0_224_quant.ckpt.index
mobilenet_v2_1.0_224_quant.ckpt.meta
mobilenet_v2_1.0_224_quant.tflite
mobilenet_v2_1.0_224_quant.tgz
mobilenet_v2_1.0_224_quant_eval.pbtxt
mobilenet_v2_1.0_224_quant_frozen.pb
mobilenet_v2_1.0_224_quant_info.txt
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS/models $
```

Ci serviranno solo il modello `mobilenet_v2_1.0_224_quant.tflite` e `labels.txt`. Si possono eliminare gli altri file.

Impostazione di Jupyter Notebook (Facoltativo)

Se si preferisce usare Jupyter Notebook per lo sviluppo:

```
pip3 install jupyter
jupyter notebook --generate-config
```

Per eseguire Jupyter Notebook, si lancia il comando (cambiare l'indirizzo IP per il proprio):

```
jupyter notebook --ip=192.168.4.210 --no-browser
```

Sul terminale, si può vedere l'indirizzo URL locale per aprire il notebook:

```
marcelo_rovai — mjrovai@raspi-zero: ~ — ssh mjrovai@192.168.4.210 — 96×12
    To access the server, open this file in a browser:
        file:///home/mjrovai/.local/share/jupyter/runtime/jpserver-7399-open.html
    Or copy and paste one of these URLs:
        http://192.168.4.210:8888/tree?token=f250f69117df5adf9d1aff01dd3ede657cb47b0ba8153447
        http://127.0.0.1:8888/tree?token=f250f69117df5adf9d1aff01dd3ede657cb47b0ba8153447
[I 2024-08-23 19:40:59.979 ServerApp] Skipped non-installed server(s): bash-language-server, dockerfile-language-server-nodejs, javascript-typescript-langserver, jedi-language-server, julia-language-server, pyright, python-language-server, python-lsp-server, r-languageserver, sql-language-server, texlab, typescript-language-server, unified-language-server, vscode-css-languageserver-bin, vscode-html-languageserver-bin, vscode-json-languageserver-bin, yaml-language-server
```

Vi si può accedere da un altro dispositivo inserendo l'indirizzo IP del Raspberry Pi e il token fornito in un browser Web (il token lo si può copiare dal terminale).

Definire la directory di lavoro nel Raspi e creare un nuovo notebook Python 3.

Verifica della Configurazione

Testare la configurazione eseguendo un semplice script Python:

```
import tflite_runtime.interpreter as tflite
import numpy as np
from PIL import Image

print("NumPy:", np.__version__)
print("Pillow:", Image.__version__)

# Try to create a TFLite Interpreter
model_path = "./models/mobilenet_v2_1.0_224_quant.tflite"
interpreter = tflite.Interpreter(model_path=model_path)
interpreter.allocate_tensors()
print("TFLite Interpreter created successfully!")
```

Si può creare lo script Python usando nano sul terminale, salvandolo con `CTRL+O` + `ENTER` + `CTRL+X`

Ed eseguirlo col comando:

Oppure lo si può lanciare direttamente sul Notebook:

Fare inferenze con Mobilenet V2

Nell'ultima sezione, abbiamo impostato l'ambiente, incluso il download di un modello pre-addestrato popolare, Mobilenet V2, addestrato sulle immagini 224x224 di ImageNet (1,2 milioni) per 1.001 classi (1.000 categorie di oggetti più 1 sfondo). Il modello è stato convertito in un formato TensorFlow Lite compatto da 3,5 MB, rendendolo adatto allo spazio di archiviazione e alla memoria limitati di un Raspberry Pi.

Apriamo un nuovo notebook per seguire tutti i passaggi per classificare un'immagine:

Importare le librerie necessarie:

```
import time
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import tflite_runtime.interpreter as tflite
```

Caricare il modello TFLite e allocare i tensori:

```
model_path = "./models/mobilenet_v2_1.0_224_quant.tflite"
interpreter = tflite.Interpreter(model_path=model_path)
interpreter.allocate_tensors()
```

Ottenere i tensori di input e output.

```
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Input details ci daranno informazioni su come il modello dovrebbe essere alimentato con un'immagine. Il profilo di (1, 224, 224, 3) ci informa che un'immagine con dimensioni (224x224x3) dovrebbe essere inserita una alla volta (Batch Dimension: 1).

```
input_details
```

```
[{'name': 'input',
  'index': 171,
  'shape': array([  1, 224, 224,   3], dtype=int32),
  'shape_signature': array([  1, 224, 224,   3], dtype=int32),
  'dtype': numpy.uint8,
  'quantization': (0.0078125, 128),
  'quantization_parameters': {'scales': array([0.0078125], dtype=float32),
   'zero_points': array([128], dtype=int32),
   'quantized_dimension': 0},
  'sparsity_parameters': {}}]
```

Input Image Shape

Gli **output details** mostrano che l'inferenza risulterà in un array di 1.001 valori interi. Tali valori derivano dalla classificazione dell'immagine, dove ogni valore è la probabilità che quella specifica etichetta sia correlata all'immagine.

```
output_details
```

```
[{'name': 'output',
  'index': 172,
  'shape': array([   1, 1001], dtype=int32),
  'shape_signature': array([   1, 1001], dtype=int32),
  'dtype': numpy.uint8,
  'quantization': (0.09889253973960876, 58),
  'quantization_parameters': {'scales': array([0.09889254], dtype=float32),
   'zero_points': array([58], dtype=int32),
   'quantized_dimension': 0},
  'sparsity_parameters': {}}]
```

Output model

Esaminiamo anche il dtype dei dettagli di input del modello

```
input_dtype = input_details[0]['dtype']
input_dtype
dtype('uint8')
```

Questo mostra che l'immagine di input dovrebbe essere composta da pixel grezzi (0 - 255). Prendiamo un'immagine di prova. La si può trasferire dal computer o scaricarne una per testarla. Per prima cosa creiamo una cartella nella nostra directory di lavoro:

```
mkdir images
cd images
wget https://upload.wikimedia.org/wikipedia/commons/3/3a/Cat03.jpg
```

Carichiamo e visualizziamo l'immagine:

```
# Load he image
img_path = "./images/Cat03.jpg"
img = Image.open(img_path)

# Display the image
plt.figure(figsize=(8, 8))
plt.imshow(img)
plt.title("Original Image")
plt.show()
```

Possiamo vedere la dimensione dell'immagine eseguendo il comando:

```
width, height = img.size
```

Questo ci mostra che l'immagine è RGB con una larghezza di 1600 e un'altezza di 1600 pixel. Quindi, per usare il nostro modello, dovremmo rimodellarlo in (224, 224, 3) e aggiungere una dimensione batch di 1, come definito nei dettagli di input: (1, 224, 224, 3). Il risultato dell'inferenza, come mostrato nei dettagli di output, sarà un array con una dimensione di 1001, come mostrato di seguito:

Quindi, rimodelliamo l'immagine, aggiungiamo la dimensione batch e vediamo il risultato:

```
img = img.resize((input_details[0]['shape'][1], input_details[0]['shape'][2]))
input_data = np.expand_dims(img, axis=0)
input_data.shape
```

La forma di input_data è come previsto: (1, 224, 224, 3)
Confermiamo il dtype dei dati di input:

```
input_data.dtype
dtype('uint8')
```

Il dtype dei dati di input è 'uint8', che è compatibile con il dtype previsto per il modello. Utilizzando input_data, eseguiamo l'interprete e otteniamo le previsioni (output):

```
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
predictions = interpreter.get_tensor(output_details[0]['index'])[0]
```

La previsione è un array con 1001 elementi. Otteniamo i primi 5 indici in cui i loro elementi hanno valori elevati:

```
top_k_results = 5
top_k_indices = np.argsort(predictions)[::-1][:top_k_results]
top_k_indices
```

top_k_indices è un array con 5 elementi: `array([283, 286, 282])`
Quindi, 283, 286, 282, 288 e 479 sono le classi più probabili dell'immagine. Avendo l'indice, dobbiamo trovare a quale classe è assegnato (ad esempio auto, gatto o cane). Il file di testo scaricato con il modello ha un'etichetta associata a ciascun indice da 0 a 1.000. Usiamo una funzione per caricare il file .txt come un elenco:

```
def load_labels(filename):
    with open(filename, 'r') as f:
        return [line.strip() for line in f.readlines()]
```

E otteniamo l'elenco, stampando le etichette associate agli indici:

```
labels_path = "./models/labels.txt"
labels = load_labels(labels_path)

print(labels[286])
print(labels[283])
print(labels[282])
print(labels[288])
print(labels[479])
```

Di conseguenza abbiamo:

```
Egyptian cat
tiger cat
tabby
lynx
carton
```

Almeno i quattro indici principali sono correlati ai felini. Il contenuto di **prediction** è la probabilità associata a ciascuna delle etichette. Come abbiamo visto nei dettagli dell'output, quei valori sono quantizzati e dovrebbero essere dequantizzati e applicare la softmax.

```
scale, zero_point = output_details[0]['quantization']
dequantized_output = (predictions.astype(np.float32) - zero_point) * scale
exp_output = np.exp(dequantized_output - np.max(dequantized_output))
probabilities = exp_output / np.sum(exp_output)
```

Stampiamo le prime 5 probabilità:

```
print (probabilities[286])
print (probabilities[283])
print (probabilities[282])
print (probabilities[288])
print (probabilities[479])
0.27741462
0.3732285
0.16919471
0.10319158
0.023410844
```

Per chiarezza, creiamo una funzione per mettere in relazione le etichette con le probabilità:

```
for i in range(top_k_results):
    print("\t{:20}: {}%".format(
        labels[top_k_indices[i]],
        (int(probabilities[top_k_indices[i]]*100))))
tiger cat           : 37%
Egyptian cat        : 27%
tabby               : 16%
lynx                : 10%
carton              : 2%
```

Definire una funzione generale di Classificazione delle Immagini

Creiamo una funzione generale per dare un'immagine come input e otteniamo le prime 5 classi possibili:

```
def image_classification(img_path, model_path, labels, top_k_results=5):
    # load the image
    img = Image.open(img_path)
    plt.figure(figsize=(4, 4))
    plt.imshow(img)
```

```
    plt.axis('off')

    # Load the TFLite model
    interpreter = tflite.Interpreter(model_path=model_path)
    interpreter.allocate_tensors()

    # Get input and output tensors
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    # Preprocess
    img = img.resize((input_details[0]['shape'][1],
                      input_details[0]['shape'][2]))
    input_data = np.expand_dims(img, axis=0)

    # Inference on Raspi-Zero
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()

    # Obtain results and map them to the classes
    predictions = interpreter.get_tensor(output_details[0]['index'])[0]

    # Get indices of the top k results
    top_k_indices = np.argsort(predictions)[::-1][:top_k_results]

    # Get quantization parameters
    scale, zero_point = output_details[0]['quantization']

    # Dequantize the output and apply softmax
    dequantized_output = (predictions.astype(np.float32) - zero_point) * scale
    exp_output = np.exp(dequantized_output - np.max(dequantized_output))
    probabilities = exp_output / np.sum(exp_output)

    print("\n\t[PREDICTION]        [Prob]\n")
    for i in range(top_k_results):
        print("\t{:20}: {}%".format(
            labels[top_k_indices[i]],
            (int(probabilities[top_k_indices[i]]*100))))
```

E caricando alcune immagini per i test, abbiamo:

Test con un modello addestrato da zero

Addestriamo un modello TFLite da zero. Per questo, si può seguire il Notebook:

CNN to classify Cifar-10 dataset

Nel notebook, abbiamo addestrato un modello utilizzando il dataset CIFAR10, che contiene 60.000 immagini da 10 classi di CIFAR (*airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck*). CIFAR ha immagini a colori 32x32 (3 canali colore) in cui gli oggetti non sono

centrati e possono avere l'oggetto con uno sfondo, come gli aerei che potrebbero avere un cielo nuvoloso dietro di loro! In breve, immagini piccole ma reali.
Il modello addestrato dalla CNN (*cifar10_model.keras*) aveva una dimensione di 2,0 MB. Utilizzando *TFLite Converter*, il modello *cifar10.tflite* è diventato di 674 MB (circa 1/3 della dimensione originale).

Sul notebook Cifar 10 - Image Classification on a Raspi with TFLite (che può essere eseguito sul Raspi), possiamo seguire gli stessi passaggi che abbiamo fatto con `mobilenet_v2_1.0_224_quant.tflite`. Di seguito sono riportati esempi di immagini che utilizzano la *General Function for Image Classification* su un Raspi-Zero, come mostrato nell'ultima sezione.

Installing Picamera2

Picamera2, una libreria Python per interagire con la fotocamera del Raspberry Pi, è basata sullo stack della fotocamera *libcamera* e la Raspberry Pi Foundation la mantiene. La libreria Picamera2 è supportata su tutti i modelli Raspberry Pi, dal Pi Zero al RPi 5. È già installata in tutto il sistema Raspi, ma dovremmo renderla accessibile all'interno dell'ambiente virtuale.

1. Per prima cosa, attivare l'ambiente virtuale se non è già attivato:

```
source ~/tflite/bin/activate
```

2. Ora, creiamo un file .pth nelll'ambiente virtuale per aggiungere il percorso del sistema del pacchetto sul sito:

```
echo "/usr/lib/python3/dist-packages" > $VIRTUAL_ENV/lib/python3.11/
site-packages/system_site_packages.pth
```

   Nota: Se la versione di Python è diversa, sostituire `python3.11` con la versione appropriata.

3. Dopo aver creato questo file, provare a importare picamera2 in Python:

```
python3
>>> import picamera2
>>> print(picamera2.__file__)
```

Il codice sopra mostrerà la posizione del file del modulo `picamera2` stesso, dimostrando che la libreria è accessibile dall'ambiente.

```
/home/mjrovai/tflite/lib/python3.11/site-packages/picamera2/__init__.py
```

È anche possibile elencare le telecamere disponibili nel sistema:

```
>>> print(Picamera2.global_camera_info())
```

Nel nostro caso, con una USB installata, si è ottenuto:

```
marcelo_rovai — mjrovai@raspi-zero: ~ — ssh mjrovai@192.168.4.210 — 116×7
>>> import picamera2
>>> print(picamera2.__file__)
/home/mjrovai/tflite/lib/python3.11/site-packages/picamera2/__init__.py
>>> print(Picamera2.global_camera_info())
[1:31:09.852601701] [6058]  INFO Camera camera_manager.cpp:313 libcamera v0.3.0+65-6ddd79b5
[{'Model': 'USB 2.0 Camera: USB Camera', 'Location': 2, 'Id': '/base/soc/usb@7e980000-1:1.0-0c45:1915', 'Num': 0}]
>>>
```

Ora che abbiamo confermato che picamera2 funziona nell'ambiente con un `indice 0`, proviamo un semplice script Python per catturare un'immagine dalla fotocamera USB:

```python
from picamera2 import Picamera2
import time

# Initialize the camera
picam2 = Picamera2() # default is index 0

# Configure the camera
config = picam2.create_still_configuration(main={"size": (640, 480)})
picam2.configure(config)

# Start the camera
picam2.start()

# Wait for the camera to warm up
time.sleep(2)

# Capture an image
picam2.capture_file("usb_camera_image.jpg")
print("Image captured and saved as 'usb_camera_image.jpg'")

# Stop the camera
picam2.stop()
```

Utilizzare l'editor di testo Nano, Jupyter Notebook o qualsiasi altro editor. Salvarlo come script Python (ad esempio, `capture_image.py`) ed eseguilo. Questo dovrebbe catturare un'immagine dalla fotocamera e salvarla come "usb_camera_image.jpg" nella stessa directory dello script.

```
marcelo_rovai — mjrovai@raspi-zero: ~/Documents/TFLITE/IMG_CLASS — ssh mjrovai@192.168.4.210 — 101×10
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS $ nano capture_image.py
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS $ ls
10_Image_Classification.ipynb  capture_image.py  models            setup_test.py
Untitled.ipynb                 images            setup_test.ipynb  usb_camera_image.jpg
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS $ python3 capture_image.py
[43:45:25.837340485] [11275]  INFO Camera camera_manager.cpp:313 libcamera v0.3.0+65-6ddd79b5
[43:45:26.077626719] [11275]  INFO Camera camera_manager.cpp:313 libcamera v0.3.0+65-6ddd79b5
[43:45:26.192062284] [11275]  INFO Camera camera.cpp:1183 configuring streams: (0) 640x480-MJPEG
Image captured and saved as 'usb_camera_image.jpg'
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/IMG_CLASS $
```

Se Jupyter è aperto, si può vedere l'immagine catturata sul computer. Altrimenti, si trasferisce il file dal Raspi al computer.

Se si sta lavorando con un Raspi-5 con un intero desktop, si può aprire il file direttamente sul dispositivo.

Progetto di Classificazione delle Immagini

Ora, svilupperemo un progetto completo di Classificazione delle Immagini utilizzando Edge Impulse Studio. Come abbiamo fatto con Movilinet V2, il modello TFLite addestrato e convertito verrà utilizzato per l'inferenza.

L'Obiettivo

Il primo passo in qualsiasi progetto ML è definire il suo obiettivo. In questo caso, è rilevare e classificare due oggetti specifici presenti in un'immagine. Per questo progetto, utilizzeremo due piccoli giocattoli: un robot e un piccolo pappagallo brasiliano (chiamato Periquito). Raccoglieremo anche immagini di un *background* in cui questi due oggetti sono assenti.

Raccolta Dati

Una volta definito l'obiettivo del nostro progetto di apprendimento automatico, il passaggio successivo, e più cruciale, è la raccolta del dataset. Possiamo utilizzare un telefono per l'acquisizione delle immagini, ma qui utilizzeremo il Raspi. Impostiamo un semplice server Web sul nostro Raspberry Pi per visualizzare le immagini `QVGA (320 x 240)` acquisite in un browser.

1. Per prima cosa, installiamo Flask, un framework Web leggero per Python:

```
pip3 install flask
```

2. Creiamo un nuovo script Python che combina l'acquisizione delle immagini con un server Web. Lo chiameremo `get_img_data.py`:

```
from flask import Flask, Response, render_template_string, request, redirect, url_for
from picamera2 import Picamera2
import io
import threading
import time
import os
import signal

app = Flask(__name__)

# Global variables
base_dir = "dataset"
picam2 = None
frame = None
frame_lock = threading.Lock()
capture_counts = {}
current_label = None
shutdown_event = threading.Event()

def initialize_camera():
    global picam2
    picam2 = Picamera2()
    config = picam2.create_preview_configuration(main={"size": (320, 240)})
    picam2.configure(config)
    picam2.start()
    time.sleep(2)  # Wait for camera to warm up

def get_frame():
    global frame
    while not shutdown_event.is_set():
        stream = io.BytesIO()
        picam2.capture_file(stream, format='jpeg')
        with frame_lock:
            frame = stream.getvalue()
        time.sleep(0.1)  # Adjust as needed for smooth preview

def generate_frames():
    while not shutdown_event.is_set():
        with frame_lock:
            if frame is not None:
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
        time.sleep(0.1)  # Adjust as needed for smooth streaming

def shutdown_server():
    shutdown_event.set()
    if picam2:
        picam2.stop()
    # Give some time for other threads to finish
    time.sleep(2)
    # Send SIGINT to the main process
    os.kill(os.getpid(), signal.SIGINT)

@app.route('/', methods=['GET', 'POST'])
def index():
    global current_label
    if request.method == 'POST':
```

```
        current_label = request.form['label']
        if current_label not in capture_counts:
            capture_counts[current_label] = 0
        os.makedirs(os.path.join(base_dir, current_label), exist_ok=True)
        return redirect(url_for('capture_page'))
    return render_template_string('''
        <!DOCTYPE html>
        <html>
        <head>
            <title>Dataset Capture - Label Entry</title>
        </head>
        <body>
            <h1>Enter Label for Dataset</h1>
            <form method="post">
                <input type="text" name="label" required>
                <input type="submit" value="Start Capture">
            </form>
        </body>
        </html>
    ''')

@app.route('/capture')
def capture_page():
    return render_template_string('''
        <!DOCTYPE html>
        <html>
        <head>
            <title>Dataset Capture</title>
            <script>
                var shutdownInitiated = false;
                function checkShutdown() {
                    if (!shutdownInitiated) {
                        fetch('/check_shutdown')
                            .then(response => response.json())
                            .then(data => {
                                if (data.shutdown) {
                                    shutdownInitiated = true;
                                    document.getElementById('video-feed').src = '';
                                    document.getElementById('shutdown-message')
                                    .style.display = 'block';
                                }

                            });
                    }

                }

                setInterval(checkShutdown, 1000);  // Check every second
            </script>
        </head>
        <body>
            <h1>Dataset Capture</h1>
            <p>Current Label: {{ label }}</p>
            <p>Images captured for this label: {{ capture_count }}</p>
            <img id="video-feed" src="{{ url_for('video_feed') }}" width="640"
            height="480" />
            <div id="shutdown-message" style="display: none; color: red;">
                Capture process has been stopped. You can close this window.
            </div>
            <form action="/capture_image" method="post">
                <input type="submit" value="Capture Image">
            </form>
            <form action="/stop" method="post">
                <input type="submit" value="Stop Capture"
                style="background-color: #ff6666;">
            </form>
            <form action="/" method="get">
                <input type="submit" value="Change Label"
                style="background-color: #ffff66;">
            </form>
        </body>
        </html>
```

```
    ''', label=current_label, capture_count=capture_counts.get(current_label, 0))

@app.route('/video_feed')
def video_feed():
    return Response(generate_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/capture_image', methods=['POST'])
def capture_image():
    global capture_counts
    if current_label and not shutdown_event.is_set():
        capture_counts[current_label] += 1
        timestamp = time.strftime("%Y%m%d-%H%M%S")
        filename = f"image_{timestamp}.jpg"
        full_path = os.path.join(base_dir, current_label, filename)

        picam2.capture_file(full_path)

    return redirect(url_for('capture_page'))

@app.route('/stop', methods=['POST'])
def stop():
    summary = render_template_string('''
        <!DOCTYPE html>
        <html>
        <head>
            <title>Dataset Capture - Stopped</title>
        </head>
        <body>
            <h1>Dataset Capture Stopped</h1>
            <p>The capture process has been stopped. You can close this window.</p>
            <p>Summary of captures:</p>
            <ul>
            {% for label, count in capture_counts.items() %}
                <li>{{ label }}: {{ count }} images</li>
            {% endfor %}
            </ul>
        </body>
        </html>
    ''', capture_counts=capture_counts)

    # Start a new thread to shutdown the server
    threading.Thread(target=shutdown_server).start()

    return summary

@app.route('/check_shutdown')
def check_shutdown():
    return {'shutdown': shutdown_event.is_set()}

if __name__ == '__main__':
    initialize_camera()
    threading.Thread(target=get_frame, daemon=True).start()
    app.run(host='0.0.0.0', port=5000, threaded=True)
```

3. Eseguire questo script:
   ```bash python3 get_img_data.py

```
4. Accedere all'interfaccia web:

    - Sul Raspberry Pi stesso (se si ha una GUI): Aprire un browser web e andare su `http
://localhost:5000`
    - Da un altro dispositivo sulla stessa rete: Aprire un browser web e andare su `http:
//<raspberry_pi_ip>:5000` (Sostituire `<raspberry_pi_ip>` con l'indirizzo IP del Raspberr
y Pi). Per esempio: `http://192.168.4.210:5000/`

Questo script Python crea un'interfaccia basata sul web per catturare e organizzare set d
i dati di immagini usando un Raspberry Pi e la sua fotocamera. È utile per progetti di ap
prendimento automatico che richiedono dati di immagini etichettati.

Caratteristiche Principali:
```
```

```
1. **Interfaccia Web**: Accessibile da qualsiasi dispositivo sulla stessa rete del Raspbe
rry Pi.
2. **Anteprima Telecamera in Tempo Reale**: Mostra un feed in tempo reale dalla telecamer
a.
3. **Sistema di Etichettatura**: Consente agli utenti di immettere etichette per diverse
categorie di immagini.
4. **Archiviazione Organizzata**: Salva automaticamente le immagini in sottodirectory spe
cifiche per etichetta.
5. **Contatori per Etichetta**: Tiene traccia di quante immagini vengono acquisite per og
ni etichetta.
6. **Statistiche di Riepilogo**: Fornisce un riepilogo delle immagini acquisite quando si
interrompe il processo di acquisizione.

#### Componenti Principali:

1. **Applicazione Web Flask**: Gestisce il routing e serve l'interfaccia Web.
2. **Integrazione di Picamera2**: Controlla la telecamera Raspberry Pi.
3. **Threaded Frame Capture**: Assicura un'anteprima live fluida.
4. **File Management**: Organizza le immagini catturate in directory etichettate.

#### Funzioni Chiave:

- `initialize_camera()`: Imposta l'istanza Picamera2.
- `get_frame()`: Cattura continuamente i frame per l'anteprima live.
- `generate_frames()`: Genera i frame per il feed video live.
- `shutdown_server()`: Imposta l'evento di arresto, arresta la telecamera e arresta il se
rver Flask
- `index()`: Gestisce la pagina di input dell'etichetta.
- `capture_page()`: Visualizza l'interfaccia di acquisizione principale.
- `video_feed()`: Mostra un'anteprima live per posizionare la telecamera
- `capture_image()`: Salva un'immagine con l'etichetta corrente.
- `stop()`: Arresta il processo di acquisizione e visualizza un riepilogo.

#### Flusso di Utilizzo:

1. Avviare lo script sul Raspberry Pi.
2. Accedere all'interfaccia web da un browser.
3. Inserire un'etichetta per le immagini da catturare e premere `Start Capture`.

![](images/png/enter_label.png)

4. Utilizzare l'anteprima live per posizionare la telecamera.
5. Cliccare `Capture Image` per salvare le immagini sotto l'etichetta corrente.

![](images/png/capture.png)

6. Cambiare le etichette come necessario per le diverse categorie, selezionando `Change L
abel`.
7. Cliccare `Stop Capture` al termine per vedere un riepilogo.

![](images/png/stop.png)

#### Note Tecniche:

- Lo script usa il threading per gestire la cattura di frame e il web serving simultanei.
- Le immagini vengono salvate con timestamp nei nomi dei file per renderle uniche.
- L'interfaccia web è reattiva e accessibile da dispositivi mobili.

#### Possibilità di Personalizzazione:

- Regolare la risoluzione dell'immagine nella funzione `initialize_camera()`. Qui abbiamo
utilizzato QVGA (320X240).
- Modificare i modelli HTML per un aspetto diverso.
- Aggiungere ulteriori passaggi di elaborazione o analisi delle immagini nella funzione `
capture_image()`.

#### Numero di campioni sul Dataset:

Si ottengono circa 60 immagini da ciascuna categoria (`periquito`, `robot` e `background`
). Provare ad acquisire con diverse angolazioni, sfondi e condizioni di luce. Sul Raspi,
finiremo con una cartella denominata `dataset`, che contiene 3 sottocartelle *periquito*,
*robot* e *background*. una per ogni classe di immagini.
```

```
Si può usare `Filezilla` per trasferire il dataset creato sul computer principale.

## Addestramento del modello con Edge Impulse

Useremo Edge Impulse Studio per addestrare il modello.  Si va nella [Pagina di Edge Impul
se](https://edgeimpulse.com/), si inseriscono le credenziali e si crea un nuovo progetto:

![](images/png/new-proj-ei.png)

> Qui si può clonare un progetto simile: [Raspi - Img Class](https://studio.edgeimpulse.c
om/public/510251/live).

### Dataset

Esamineremo quattro passaggi principali usando EI Studio (o Studio). Questi passaggi sono
fondamentali per preparare il nostro modello per l'uso sul Raspi: Dataset, Impulse, Test
e Deploy (sul dispositivo Edge, in questo caso, il Raspi).

> Per quanto riguarda il Dataset, è essenziale sottolineare che il nostro Dataset origina
le, acquisito con il Raspi, sarà suddiviso in *Training*, *Validation* e *Test*. Il Test
Set sarà separato dall'inizio e riservato per l'uso solo nella fase di Test dopo l'addest
ramento. Il Validation Set sarà utilizzato durante l'addestramento.

Su Studio, seguire i passaggi per caricare i dati acquisiti:

1. Si va nella scheda `Data acquisition` e nella sezione `UPLOAD DATA`, si caricano i fil
e dal computer nelle categorie scelte.
2. Lasciare a Studio la suddivisione del dataset originale in *train and test* e sceglier
e l'etichetta a riguardo
3. Ripetere la procedura per tutte e tre le classi. Alla fine, si vedranno i "raw data" i
n Studio:

![](images/png/data-Aquisition.png)

Studio consente di esplorare i dati, mostrando una vista completa di tutti i quelli nel p
rogetto. Si possono cancellare, ispezionare o modificare le etichette cliccando sui singo
li elementi di dati. Nel nostro caso, un progetto semplice, i dati sembrano OK.

![](images/png/data-esplorer.png)

## Il Progetto Impulse

In questa fase, dovremmo definire come:

- Pre-elaborare i nostri dati, il che consiste nel ridimensionare le singole immagini e d
eterminare la `color depth` [profondità di colore] da utilizzare (sia RGB che in scala di
grigi) e

- Specificare un Modello. In questo caso, sarà `Transfer Learning (Images)` a mettere a p
unto un modello di classificazione delle immagini MobileNet V2 pre-addestrato sui nostri
dati. Questo metodo funziona bene anche con set di dati di immagini relativamente piccoli
(circa 180 immagini nel nostro caso).

Transfer Learning con MobileNet offre un approccio semplificato all'addestramento del mod
ello, che è particolarmente utile per ambienti con risorse limitate e progetti con dati e
tichettati limitati. MobileNet, noto per la sua architettura leggera, è un modello pre-ad
destrato che ha già appreso funzionalità preziose da un ampio set di dati (ImageNet).

![](images/jpeg/model_1.jpg)

Sfruttando queste funzionalità apprese, possiamo addestrare un nuovo modello per il compi
to specifico con meno dati e risorse computazionali e raggiungere un'accuratezza competit
iva.

![](images/jpeg/model_2.jpg)

Questo approccio riduce significativamente i tempi di addestramento e i costi computazion
ali, rendendolo ideale per la prototipazione rapida e l'implementazione su dispositivi em
bedded in cui l'efficienza è fondamentale.

Si va alla scheda Impulse Design e si crea l'*impulse*, definendo una dimensione dell'imm
```

```
agine di 160x160 e schiacciandola (forma quadrata, senza ritaglio). Si seleziona Image e
i blocchi Transfer Learning. Si salva l'Impulse.

![](images/png/impulse.png)

### Pre-elaborazione delle immagini

Tutte le immagini QVGA/RGB565 in ingresso verranno convertite in 76.800 feature (160x160x
3).

![](images/png/preproc.png)

Premere `Save parameters` e selezionare `Generate features` nella scheda successiva.

### Progettazione del modello

MobileNet è una famiglia di reti neurali convoluzionali efficienti progettate per applica
zioni di visione mobile e embedded. Le caratteristiche principali di MobileNet sono:

1. Leggero: Ottimizzato per dispositivi mobili e sistemi embedded con risorse di calcolo
limitate.
2. Velocità: Tempi di inferenza rapidi, adatti per applicazioni in tempo reale.
3. Precisione: Mantiene una buona accuratezza nonostante le dimensioni compatte.

[MobileNetV2](https://arxiv.org/abs/1801.04381), introdotto nel 2018, migliora l'architet
tura MobileNet originale. Le caratteristiche principali includono:

1. Residui Invertiti: Le strutture residue invertite vengono utilizzate quando vengono cr
eate connessioni di scelta rapida tra layer di colli di bottiglia sottili.
2. Colli di Bottiglia Lineari: Rimuove le non linearità nei layer stretti per impedire la
distruzione delle informazioni.
3. Convoluzioni Separabili in Profondità: Continua a utilizzare questa efficiente operazi
one da MobileNetV1.

Nel nostro progetto, faremo un `Transfer Learning` con `MobileNetV2 160x160 1.0`, il che
significa che le immagini utilizzate per l'addestramento (e l'inferenza futura) dovrebber
o avere una *input Size* [dimensione di input] di 160x160 pixel e un *Width Multiplier* [
moltiplicatore di larghezza] di 1.0 (larghezza completa, non ridotta). Questa configurazi
one bilancia tra dimensione del modello, velocità e accuratezza.

### Training del Modello

Un'altra preziosa tecnica di apprendimento profondo è il **Data Augmentation**. Il "data
augmentation" migliora l'accuratezza dei modelli di apprendimento automatico creando dati
artificiali aggiuntivi. Un sistema di data augmentation apporta piccole modifiche casuali
ai dati di training durante l'addestramento (ad esempio capovolgendo, ritagliando o ruota
ndo le immagini).

Guardando internamente, qui si può vedere come Edge Impulse implementa una policy di data
Augmentation sui dati:

``` python
# Implements the data augmentation policy
def augment_image(image, label):
    # Flips the image randomly
    image = tf.image.random_flip_left_right(image)

    # Increase the image size, then randomly crop it down to
    # the original dimensions
    resize_factor = random.uniform(1, 1.2)
    new_height = math.floor(resize_factor * INPUT_SHAPE[0])
    new_width = math.floor(resize_factor * INPUT_SHAPE[1])
    image = tf.image.resize_with_crop_or_pad(image, new_height, new_width)
    image = tf.image.random_crop(image, size=INPUT_SHAPE)

    # Vary the brightness of the image
    image = tf.image.random_brightness(image, max_delta=0.2)

    return image, label
```
```

L'esposizione a queste variazioni durante l'addestramento può aiutare a impedire al modello di prendere scorciatoie "memorizzando" indizi superficiali nei dati di addestramento, il che significa che potrebbe riflettere meglio i pattern profondi in esame nel set di dati.
L'ultimo layer denso del nostro modello avrà 0 neuroni con un dropout del 10% per prevenire il sovradattamento. Ecco il risultato del Training:

Il risultato è eccellente, con una latenza ragionevole di 35 ms (per un Raspi-4), che dovrebbe tradursi in circa 30 fps (fotogrammi al secondo) durante l'inferenza. Un Raspi-Zero dovrebbe essere più lento e il Raspi-5 più veloce.

Compromesso: Accuratezza contro Velocità

Se è necessaria un'inferenza più rapida, dovremmo addestrare il modello usando alfa più piccoli (0.35, 0.5 e 0.75) o persino ridurre le dimensioni dell'immagine in ingresso, a discapito dell'accuratezza. Tuttavia, ridurre le dimensioni dell'immagine in ingresso e diminuire l'alfa (moltiplicatore di larghezza) può accelerare l'inferenza per MobileNet V2, ma hanno compromessi diversi. Confrontiamoli:

1. Riduzione delle Dimensioni dell'Immagine in Ingresso:

Pro:

- Riduce significativamente il costo computazionale su tutti i layer.
- Riduce l'utilizzo della memoria.
- Spesso fornisce un aumento sostanziale della velocità.

Contro:

- Potrebbe ridurre la capacità del modello di rilevare piccole caratteristiche o dettagli fini.
- Può avere un impatto significativo sulla precisione, specialmente per le attività che richiedono un riconoscimento a grana fine.

2. Riduzione di Alpha (Moltiplicatore di Larghezza):

Pro:

- Riduce il numero di parametri e calcoli nel modello.
- Mantiene la risoluzione di input originale, preservando potenzialmente più dettagli.
- Può fornire un buon equilibrio tra velocità e precisione.

Contro:

- Potrebbe non accelerare l'inferenza in modo così drastico come la riduzione delle dimensioni di input.
- Può ridurre la capacità del modello di apprendere caratteristiche complesse.

Confronto:

1. Impatto sulla Velocità:
    - La riduzione delle dimensioni di input spesso fornisce un aumento di velocità più sostanziale perché riduce i calcoli in modo quadratico (dimezzando sia la larghezza che l'altezza si riducono i calcoli di circa il 75%).
    - La riduzione di Alpha fornisce una riduzione più lineare nei calcoli.
2. Impatto sulla Precisione:
    - La riduzione delle dimensioni di input può avere un impatto significativo sulla precisione, specialmente quando si rilevano piccoli oggetti o dettagli fini.
    - La riduzione di alpha tende ad avere un impatto più graduale sulla precisione.
3. Architettura del Modello:
    - La modifica delle dimensioni di input non altera l'architettura del modello.
    - La modifica di alpha modifica la struttura del modello riducendo il numero di canali in ogni layer.

Raccomandazione:

1. Se l'applicazione non richiede il rilevamento di piccoli dettagli e può tollerare una certa perdita di accuratezza, ridurre le dimensioni di input è spesso il modo più efficace per accelerare l'inferenza.
2. Ridurre l'alfa potrebbe essere preferibile se mantenere la capacità di rilevare dettagli fini è fondamentale o se c'è bisogno di un compromesso più equilibrato tra velocità e accuratezza.
3. Per ottenere risultati migliori, si devono sperimentare entrambi:
    - Provare MobileNet V2 con dimensioni di input come 160x160 o 92x92
    - Sperimentare con valori alfa come 1.0, 0.75, 0.5 o 0.35.
4. Eseguire sempre il benchmark delle diverse configurazioni sull'hardware specifico e con il particolare set di dati per trovare l'equilibrio ottimale per il caso d'uso.

Ricordarsi che la scelta migliore dipende dai requisiti specifici di accuratezza, velocità e dalla natura delle immagini con cui si sta lavorando. Spesso vale la pena sperimentare combinazioni per trovare la configurazione ottimale per il particolare caso d'uso.

Test del Modello

Ora, si deve prendere il set di dati all'inizio del progetto ed eseguire il modello addestrato usandolo come input. Di nuovo, il risultato è eccellente (92,22%).

Distribuzione del modello

Come abbiamo fatto nella sezione precedente, possiamo distribuire il modello addestrato come .tflite e usare Raspi per eseguirlo usando Python.
Nella scheda `Dashboard`, si va su Transfer learning model (int8 quantized) e si clicca sull'icona di download:

Download block output

TITLE	TYPE	SIZE
Image training data	NPY file	150 windows
Image training labels	NPY file	150 windows
Image testing data	NPY file	36 windows
Image testing labels	NPY file	36 windows
Transfer learning model	TensorFlow Lite (float32)	9 MB
Transfer learning model	TensorFlow Lite (int8 quantized)	3 MB
Transfer learning model	Model evaluation metrics (JSON file)	5 KB
Transfer learning model	TensorFlow SavedModel	8 MB
Transfer learning model	Keras h5 model	8 MB

Scarichiamo anche la versione float32 per il confronto

Trasferire il modello dal computer al Raspi (./models), ad esempio, usando FileZilla. Catturare, inoltre, alcune immagini per l'inferenza (./images).
Importare le librerie necessarie:

```
import time
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import tflite_runtime.interpreter as tflite
```

Definire i path e le etichette:

```
img_path = "./images/robot.jpg"
model_path = "./models/ei-raspi-img-class-int8-quantized-model.tflite"
labels = ['background', 'periquito', 'robot']
```

Notare che i modelli addestrati su Edge Impulse Studio produrranno valori con indice 0, 1, 2, ecc., dove le etichette effettive seguiranno un ordine alfabetico.

Caricare il modello, allocare i tensori e ottenere i dettagli dei tensori di input e output:

```
# Load the TFLite model
interpreter = tflite.Interpreter(model_path=model_path)
interpreter.allocate_tensors()

# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Una differenza importante da notare è che il `dtype` dei dettagli di input del modello è ora `int8`, il che significa che i valori di input vanno da -128 a +127, mentre ogni pixel della nostra immagine va da 0 a 255. Ciò significa che dovremmo pre-elaborare l'immagine per farla corrispondere. Possiamo controllare qui:

```
input_dtype = input_details[0]['dtype']
input_dtype
numpy.int8
```

Quindi, apriamo l'immagine e mostriamola:

```
img = Image.open(img_path)
plt.figure(figsize=(4, 4))
plt.imshow(img)
plt.axis('off')
plt.show()
```

Ed eseguiamo la pre-elaborazione:

```
scale, zero_point = input_details[0]['quantization']
img = img.resize((input_details[0]['shape'][1],
                  input_details[0]['shape'][2]))
img_array = np.array(img, dtype=np.float32) / 255.0
img_array = (img_array / scale + zero_point).clip(-128, 127).astype(np.int8)
input_data = np.expand_dims(img_array, axis=0)
```

Controllando i dati di input, possiamo verificare che il tensore di input è compatibile con quanto previsto dal modello:

```
input_data.shape, input_data.dtype
((1, 160, 160, 3), dtype('int8'))
```

Adesso è il momento di effettuare l'inferenza. Calcoliamo anche la latenza del modello:

```
# Inference on Raspi-Zero
start_time = time.time()
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
end_time = time.time()
inference_time = (end_time - start_time) * 1000  # Convert to milliseconds
print ("Inference time: {:.1f}ms".format(inference_time))
```

Il modello impiegherà circa 125 ms per eseguire l'inferenza nel Raspi-Zero, che dura 3 o 4 volte più a lungo di un Raspi-5.

Ora possiamo ottenere le etichette di output e le probabilità. È anche importante notare che il modello addestrato su Edge Impulse Studio ha un softmax nel suo output (diverso dal Movilenet V2 originale) e dovremmo usare l'output grezzo del modello come "probabilità".

```
# Obtain results and map them to the classes
predictions = interpreter.get_tensor(output_details[0]['index'])[0]

# Get indices of the top k results
top_k_results=3
top_k_indices = np.argsort(predictions)[::-1][:top_k_results]

# Get quantization parameters
```

```
scale, zero_point = output_details[0]['quantization']

# Dequantize the output
dequantized_output = (predictions.astype(np.float32) - zero_point) * scale
probabilities = dequantized_output

print("\n\t[PREDICTION]        [Prob]\n")
for i in range(top_k_results):
    print("\t{:20}: {:.2f}%".format(
        labels[top_k_indices[i]],
        probabilities[top_k_indices[i]] * 100))
```

```
[PREDICTION]        [Prob]

robot               : 99.61%
periquito           : 0.00%
background          : 0.00%
```

Modifichiamo la funzione creata in precedenza in modo da poter gestire diversi tipi di modelli:

```
def image_classification(img_path, model_path, labels, top_k_results=3,
                         apply_softmax=False):
    # Load the image
    img = Image.open(img_path)
    plt.figure(figsize=(4, 4))
    plt.imshow(img)
    plt.axis('off')

    # Load the TFLite model
    interpreter = tflite.Interpreter(model_path=model_path)
    interpreter.allocate_tensors()

    # Get input and output tensors
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    # Preprocess
    img = img.resize((input_details[0]['shape'][1],
                      input_details[0]['shape'][2]))

    input_dtype = input_details[0]['dtype']

    if input_dtype == np.uint8:
        input_data = np.expand_dims(np.array(img), axis=0)
    elif input_dtype == np.int8:
        scale, zero_point = input_details[0]['quantization']
        img_array = np.array(img, dtype=np.float32) / 255.0
        img_array = (img_array / scale + zero_point).clip(-128, 127).astype(np.int8)
        input_data = np.expand_dims(img_array, axis=0)
    else:  # float32
        input_data = np.expand_dims(np.array(img, dtype=np.float32), axis=0) / 255.0

    # Inference on Raspi-Zero
    start_time = time.time()
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    end_time = time.time()
    inference_time = (end_time - start_time) * 1000  # Convert to milliseconds

    # Obtain results
    predictions = interpreter.get_tensor(output_details[0]['index'])[0]

    # Get indices of the top k results
    top_k_indices = np.argsort(predictions)[::-1][:top_k_results]

    # Handle output based on type
    output_dtype = output_details[0]['dtype']
    if output_dtype in [np.int8, np.uint8]:
        # Dequantize the output
        scale, zero_point = output_details[0]['quantization']
```

```
        predictions = (predictions.astype(np.float32) - zero_point) * scale

    if apply_softmax:
        # Apply softmax
        exp_preds = np.exp(predictions - np.max(predictions))
        probabilities = exp_preds / np.sum(exp_preds)
    else:
        probabilities = predictions

    print("\n\t[PREDICTION]        [Prob]\n")
    for i in range(top_k_results):
        print("\t{:20}: {:.1f}%".format(
            labels[top_k_indices[i]],
            probabilities[top_k_indices[i]] * 100))
    print ("\n\tInference time: {:.1f}ms".format(inference_time))
```

E lo si testa con immagini diverse e con il modello quantizzato int8 (**160x160 alpha =1.0**).

Scarichiamo un modello più piccolo, come quello addestrato per il Nicla Vision Lab (int8 quantized model, 96x96, alpha = 0.1), come test. Possiamo usare la stessa funzione:

Il modello ha perso un po' di accuratezza, ma è ancora OK dato che non cerca molti dettagli. Per quanto riguarda la latenza, siamo circa **dieci volte più veloci** su Raspi-Zero.

Classificazione delle Immagini in Tempo Reale

Sviluppiamo un'app per catturare immagini con la fotocamera USB in tempo reale, mostrandone la classificazione.

Utilizzando nano sul terminale, salvare il codice sottostante, come `img_class_live_infer.py`.

```
from flask import Flask, Response, render_template_string, request, jsonify
from picamera2 import Picamera2
import io
import threading
import time
import numpy as np
from PIL import Image
```

```
import tflite_runtime.interpreter as tflite
from queue import Queue

app = Flask(__name__)

# Global variables
picam2 = None
frame = None
frame_lock = threading.Lock()
is_classifying = False
confidence_threshold = 0.8
model_path = "./models/ei-raspi-img-class-int8-quantized-model.tflite"
labels = ['background', 'periquito', 'robot']
interpreter = None
classification_queue = Queue(maxsize=1)

def initialize_camera():
    global picam2
    picam2 = Picamera2()
    config = picam2.create_preview_configuration(main={"size": (320, 240)})
    picam2.configure(config)
    picam2.start()
    time.sleep(2)  # Wait for camera to warm up

def get_frame():
    global frame
    while True:
        stream = io.BytesIO()
        picam2.capture_file(stream, format='jpeg')
        with frame_lock:
            frame = stream.getvalue()
        time.sleep(0.1)  # Capture frames more frequently

def generate_frames():
    while True:
        with frame_lock:
            if frame is not None:
                yield (b'--frame\r\n'
                       b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
        time.sleep(0.1)

def load_model():
    global interpreter
    if interpreter is None:
        interpreter = tflite.Interpreter(model_path=model_path)
        interpreter.allocate_tensors()
    return interpreter

def classify_image(img, interpreter):
    input_details = interpreter.get_input_details()
    output_details = interpreter.get_output_details()

    img = img.resize((input_details[0]['shape'][1],
                      input_details[0]['shape'][2]))
    input_data = np.expand_dims(np.array(img), axis=0)\
                            .astype(input_details[0]['dtype'])

    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()

    predictions = interpreter.get_tensor(output_details[0]['index'])[0]
    # Handle output based on type
    output_dtype = output_details[0]['dtype']
    if output_dtype in [np.int8, np.uint8]:
        # Dequantize the output
        scale, zero_point = output_details[0]['quantization']
        predictions = (predictions.astype(np.float32) - zero_point) * scale
    return predictions

def classification_worker():
    interpreter = load_model()
    while True:
```

```
        if is_classifying:
            with frame_lock:
                if frame is not None:
                    img = Image.open(io.BytesIO(frame))
            predictions = classify_image(img, interpreter)
            max_prob = np.max(predictions)
            if max_prob >= confidence_threshold:
                label = labels[np.argmax(predictions)]
            else:
                label = 'Uncertain'
            classification_queue.put({'label': label,
                                      'probability': float(max_prob)})
        time.sleep(0.1)  # Adjust based on your needs

@app.route('/')
def index():
    return render_template_string('''
        <!DOCTYPE html>
        <html>
        <head>
            <title>Image Classification</title>
            <script
                src="https://code.jquery.com/jquery-3.6.0.min.js">
            </script>
            <script>
                function startClassification() {
                    $.post('/start');
                    $('#startBtn').prop('disabled', true);
                    $('#stopBtn').prop('disabled', false);
                }

                function stopClassification() {
                    $.post('/stop');
                    $('#startBtn').prop('disabled', false);
                    $('#stopBtn').prop('disabled', true);
                }

                function updateConfidence() {
                    var confidence = $('#confidence').val();
                    $.post('/update_confidence', {confidence: confidence});
                }

                function updateClassification() {
                    $.get('/get_classification', function(data) {
                        $('#classification').text(data.label + ': '
                        + data.probability.toFixed(2));
                    });
                }

                $(document).ready(function() {
                    setInterval(updateClassification, 100);
                    // Update every 100ms
                });
            </script>
        </head>
        <body>
            <h1>Image Classification</h1>
            <img src="{{ url_for('video_feed') }}" width="640" height="480" />
            <br>
            <button id="startBtn" onclick="startClassification()">
            Start Classification</button>
            <button id="stopBtn" onclick="stopClassification()" disabled>
            Stop Classification</button>
            <br>
            <label for="confidence">Confidence Threshold:</label>
            <input type="number" id="confidence" name="confidence" min="0"
            max="1" step="0.1" value="0.8" onchange="updateConfidence()">
            <br>
            <div id="classification">Waiting for classification...</div>
        </body>
        </html>
    ''')
```

```
@app.route('/video_feed')
def video_feed():
    return Response(generate_frames(),
                    mimetype='multipart/x-mixed-replace; boundary=frame')

@app.route('/start', methods=['POST'])
def start_classification():
    global is_classifying
    is_classifying = True
    return '', 204

@app.route('/stop', methods=['POST'])
def stop_classification():
    global is_classifying
    is_classifying = False
    return '', 204

@app.route('/update_confidence', methods=['POST'])
def update_confidence():
    global confidence_threshold
    confidence_threshold = float(request.form['confidence'])
    return '', 204

@app.route('/get_classification')
def get_classification():
    if not is_classifying:
        return jsonify({'label': 'Not classifying', 'probability': 0})
    try:
        result = classification_queue.get_nowait()
    except Queue.Empty:
        result = {'label': 'Processing', 'probability': 0}
    return jsonify(result)

if __name__ == '__main__':
    initialize_camera()
    threading.Thread(target=get_frame, daemon=True).start()
    threading.Thread(target=classification_worker, daemon=True).start()
    app.run(host='0.0.0.0', port=5000, threaded=True)
```

Sul terminale lanciare:

```
python3 img_class_live_infer.py
```

E accedere all'interfaccia web:

- Sul Raspberry Pi stesso (se si ha una GUI): si apre un browser web e si va su`http://localhost:5000`
- Da un altro dispositivo sulla stessa rete: aprire un browser web e andare su `http://<raspberry_pi_ip>:5000` (Sostituire `<raspberry_pi_ip>` con l'indirizzo IP del Raspberry Pi). Per esempio: `http://192.168.4.210:5000/`

Ecco alcuni screenshot dell'app in esecuzione su un desktop esterno

Qui si può vedere l'app in esecuzione su YouTube:
https://www.youtube.com/watch?v=o1QsQrpCMw4

Il codice crea un'applicazione web per la classificazione delle immagini in tempo reale utilizzando un Raspberry Pi, il suo modulo fotocamera e un modello TensorFlow Lite. L'applicazione utilizza Flask per fornire un'interfaccia web in cui è possibile visualizzare il feed della fotocamera e vedere i risultati della classificazione in tempo reale.

Componenti Chiave:

1. **Applicazione Web Flask**: Fornisce l'interfaccia utente e gestisce le richieste.
2. **PiCamera2**: Cattura le immagini dal modulo fotocamera Raspberry Pi.
3. **TensorFlow Lite**: Esegue il modello di classificazione delle immagini.
4. **Threading**: Gestisce le operazioni simultanee per prestazioni fluide.

Caratteristiche Principali:

- Visualizzazione feed telecamera live
- Classificazione immagini in tempo reale
- Soglia di confidenza regolabile
- Avvia/Arresta classificazione su richiesta

Struttura del Codice:

1. **Importazioni e Setup**:
   - Flask per applicazione web
   - PiCamera2 per controllo telecamera
   - TensorFlow Lite per l'inferenza
   - Threading e Queue per operazioni concorrenti
2. **Variabili Globali**:
   - Gestione telecamera e frame
   - Controllo classificazione
   - Informazioni modello ed etichetta
3. **Funzioni della Telecamera**:
   - `initialize_camera()`: Imposta PiCamera2
   - `get_frame()`: Cattura continuamente frame
   - `generate_frames()`: Genera frame per il feed web
4. **Funzioni del Modello**:
   - `load_model()`: Carica il modello TFLite
   - `classify_image()`: Esegue l'inferenza su una singola immagine
5. **Worker per la Classificazione**:
   - Gira in un thread separato
   - Classifica continuamente i frame quando è attivo
   - Aaggiorna una coda con i risultati più recenti
6. **Route di Flask**:
   - `/`: Serve la pagina HTML principale
   - `/video_feed`: Trasmette il feed della telecamera
   - `/start` and `/stop`: Controlla la classificazione
   - `/update_confidence`: Regola la soglia di confidenza
   - `/get_classification`: Restituisce l'ultimo risultato di classificazione
7. **Template HTML**:
   - Visualizza il feed della telecamera e la classificazione dei risultati
   - Fornisce controlli per avviare/arrestare e regolare le impostazioni

8. **Esecuzione Principale**:
    - Inizializza la fotocamera e avvia i thread necessari
    - Esegue l'applicazione Flask

Concetti Chiave:

1. **Operazioni Concorrenti**: Utilizzo di thread per gestire l'acquisizione e la classificazione della telecamera separatamente dal server Web.
2. **Aggiornamenti in Tempo Reale**: Aggiornamenti frequenti dei risultati della classificazione senza ricaricare la pagina.
3. **Riutilizzo del Modello**: Caricamento del modello TFLite una volta e il riutilizzo per l'efficienza.
4. **Configurazione Flessibile**: Consente agli utenti di regolare la soglia di confidenza al volo.

Uso:

1. Assicurarsi che tutte le dipendenze siano installate.
2. Eseguire lo script su un Raspberry Pi con un modulo telecamera.
3. Accedere all'interfaccia Web da un browser utilizzando l'indirizzo IP del Raspberry Pi.
4. Avviare la classificazione e regolare le impostazioni in base alle esigenze.

Conclusione:

La classificazione delle immagini è emersa come un'applicazione potente e versatile dell'apprendimento automatico, con implicazioni significative per vari campi, dall'assistenza sanitaria al monitoraggio ambientale. Questo capitolo ha dimostrato come implementare un sistema di classificazione delle immagini robusto su dispositivi edge come Raspi-Zero e Raspi-5, mostrando il potenziale per l'intelligenza in tempo reale sul dispositivo.
Abbiamo esplorato l'intera pipeline di un progetto di classificazione delle immagini, dalla raccolta dati e dall'addestramento del modello tramite Edge Impulse Studio all'implementazione e all'esecuzione di inferenze su un Raspi. Il processo ha evidenziato diversi punti chiave:

1. L'importanza di una corretta raccolta dati e pre-elaborazione per l'addestramento efficace dei modelli.
2. La potenza dell'apprendimento tramite trasferimento, che ci consente di sfruttare modelli pre-addestrati come MobileNet V2 per un addestramento efficiente con dati limitati.
3. I compromessi tra accuratezza del modello e velocità di inferenza, particolarmente cruciali per i dispositivi edge.
4. L'implementazione della classificazione in tempo reale tramite un'interfaccia basata sul Web, che dimostra applicazioni pratiche.

La capacità di eseguire questi modelli su dispositivi edge come Raspi apre numerose possibilità per applicazioni IoT, sistemi autonomi e soluzioni di monitoraggio in tempo reale. Consente una latenza ridotta, una migliore privacy e il funzionamento in ambienti con connettività limitata.
Come abbiamo visto, anche con i vincoli computazionali dei dispositivi edge, è possibile ottenere risultati impressionanti in termini di accuratezza e velocità. La flessibilità di regolare i parametri del modello, come le dimensioni di input e i valori alfa, consente una messa a punto precisa per soddisfare requisiti di progetto specifici.
Guardando al futuro, il campo dell'intelligenza artificiale edge e della classificazione delle immagini continua a evolversi rapidamente. I progressi nelle tecniche di compressione dei

modelli, nell'accelerazione hardware e nelle architetture di reti neurali più efficienti promettono di espandere ulteriormente le capacità dei dispositivi edge nelle attività di visione artificiale.
Questo progetto funge da base per applicazioni di visione artificiale più complesse e incoraggia un'ulteriore esplorazione nell'entusiasmante mondo dell'intelligenza artificiale edge e dell'IoT. Che si tratti di automazione industriale, applicazioni per la casa intelligente o monitoraggio ambientale, le competenze e i concetti trattati qui forniscono un solido punto di partenza per un'ampia gamma di progetti innovativi.

Risorse

- Esempio di Dataset
- Impostazione del Notebook di Prova su un Raspi
- Notebook di Classificazione delle Immagini su un Raspi
- CNN per classificare il dataset Cifar-10 su CoLab
- Cifar 10 - Classificazione delle Immagini su un Raspi
- Script Python
- Progetto Edge Impulse

Rilevamento degli Oggetti

DALL·E prompt - Immagine di copertina per un capitolo "Rilevamento oggetti" in un tutorial su Raspberry Pi, progettata nello stesso stile da laboratorio di elettronica degli anni '50 delle copertine precedenti. La scena dovrebbe presentare in primo piano ruote e cubi, simili a quelli forniti dall'utente, posizionati su un banco da lavoro. Un Raspberry Pi con un modulo fotocamera collegato dovrebbe catturare un'immagine di questi oggetti. Circondare la scena con classici strumenti da laboratorio come saldatori, resistori e fili. Lo sfondo del laboratorio dovrebbe includere apparecchiature d'epoca come oscilloscopi e radio a valvole, mantenendo il tocco dettagliato e nostalgico dell'epoca. Non dovrebbero essere inclusi testo o loghi.

Introduzione

Sulla base della nostra esplorazione della classificazione delle immagini, ora rivolgiamo la nostra attenzione a un'attività di visione artificiale più avanzata: il rilevamento degli oggetti. Mentre la classificazione delle immagini assegna una singola etichetta a un'intera immagine, il rilevamento degli oggetti va oltre identificando e localizzando più oggetti all'interno di una singola immagine. Questa capacità apre molte nuove applicazioni e sfide, in particolare nell'edge computing e nei dispositivi IoT come Raspberry Pi.

Il rilevamento degli oggetti combina le attività di classificazione e localizzazione. Non solo determina quali oggetti sono presenti in un'immagine, ma individua anche le loro posizioni, ad esempio disegnando riquadri di delimitazione attorno a essi. Questa complessità aggiunta

rende il rilevamento degli oggetti uno strumento più potente per comprendere le scene visive, ma richiede anche modelli e tecniche di training più sofisticati.
Nell'IA edge, dove lavoriamo con risorse computazionali limitate, l'implementazione di modelli di rilevamento degli oggetti efficienti diventa cruciale. Le sfide che abbiamo affrontato con la classificazione delle immagini, ovvero bilanciare le dimensioni del modello, la velocità di inferenza e l'accuratezza, sono amplificate nel rilevamento degli oggetti. Tuttavia, i vantaggi sono anche più significativi, poiché il rilevamento degli oggetti consente un'analisi dei dati visivi più sfumata e dettagliata.
Alcune applicazioni del rilevamento di oggetti su dispositivi edge includono:

1. Sistemi di sorveglianza e sicurezza
2. Veicoli autonomi e droni
3. Controllo industriale della qualità
4. Monitoraggio della fauna selvatica
5. Applicazioni di realtà aumentata

Mettendo le mani nel rilevamento di oggetti, ci baseremo sui concetti e sulle tecniche esplorate nella classificazione delle immagini. Esamineremo le architetture di rilevamento di oggetti più diffuse progettate per l'efficienza, come:

- "Single Stage Detectors" [Rilevatori a stadio singolo], come MobileNet ed EfficientDet,
- FOMO (Faster Objects, More Objects), e
- YOLO (You Only Look Once).

Per saperne di più sui modelli di rilevamento di oggetti, seguire il tutorial A Gentle Introduction to Object Recognition With Deep Learning.

Esploreremo quei modelli di rilevamento di oggetti utilizzando

- TensorFlow Lite Runtime (ora modificato in LiteRT),
- Edge Impulse Linux Python SDK e
- Ultralitics

In questo laboratorio, tratteremo i fondamenti del rilevamento degli oggetti e le sue differenze rispetto alla classificazione delle immagini. Impareremo anche come addestrare, mettere a punto, testare, ottimizzare e distribuire le architetture di rilevamento degli oggetti più diffuse utilizzando un dataset creato da zero.

Fondamenti dell'Object Detection

Il rilevamento degli oggetti si basa sulle fondamenta della classificazione delle immagini, ma ne amplia notevolmente le capacità. Per comprendere il rilevamento degli oggetti, è fondamentale innanzitutto riconoscere le sue principali differenze rispetto alla classificazione delle immagini:

Classificazione delle Immagini vs. Rilevamento degli Oggetti

Classificazione delle Immagini:

- Assegna una singola etichetta a un'intera immagine
- Risponde alla domanda: "Qual è l'oggetto o la scena principale di questa immagine?"
- Genera una singola previsione di classe per l'intera immagine

Rilevamento degli oggetti:

- Identifica e individua più oggetti all'interno di un'immagine
- Risponde alle domande: "Quali oggetti sono presenti in questa immagine e dove si trovano?"
- Genera più previsioni, ciascuna composta da un'etichetta di classe e un riquadro di delimitazione

Per visualizzare questa differenza, prendiamo in considerazione un esempio:

Questo diagramma illustra la differenza critica: la classificazione delle immagini fornisce un'unica etichetta per l'intera immagine, mentre il rilevamento degli oggetti identifica più oggetti, le loro classi e le loro posizioni all'interno dell'immagine.

Componenti Chiave del Rilevamento degli Oggetti

I sistemi di rilevamento degli oggetti sono in genere costituiti da due componenti principali:

1. Localizzazione degli Oggetti: Questo componente identifica dove si trovano gli oggetti nell'immagine. In genere genera riquadri di delimitazione, regioni rettangolari che racchiudono ogni oggetto rilevato.
2. Classificazione degli Oggetti: Questo componente determina la classe o la categoria di ogni oggetto rilevato, in modo simile alla classificazione delle immagini ma applicato a ogni regione localizzata.

Sfide nel Rilevamento degli Oggetti

Il rilevamento degli oggetti presenta diverse sfide oltre a quelle della classificazione delle immagini:

- Oggetti multipli: Un'immagine può contenere più oggetti di varie classi, dimensioni e posizioni.
- Scale variabili: Gli oggetti possono apparire a diverse dimensioni all'interno dell'immagine.
- Occlusione: Gli oggetti possono essere parzialmente nascosti o sovrapposti.
- Disordine di sfondo: Distinguere gli oggetti da sfondi complessi può essere difficile.
- Prestazioni in tempo reale: Molte applicazioni richiedono tempi di inferenza rapidi, soprattutto su dispositivi edge.

Approcci al Rilevamento di Oggetti

Esistono due approcci principali al rilevamento di oggetti:

1. Rilevatori a due stadi: Prima propongono le regioni di interesse e poi classificano ciascuna regione. Esempi includono R-CNN e le sue varianti (Fast R-CNN, Faster R-CNN).
2. Rilevatori a stadio singolo: Questi prevedono bounding box (o centroidi) e probabilità di classe in un passaggio "forward" [in avanti] della rete. Esempi includono YOLO (You Only Look Once), EfficientDet, SSD (Single Shot Detector) e FOMO (Faster Objects, More Objects). Questi sono spesso più veloci e più adatti per dispositivi edge come Raspberry Pi.

Metriche di Valutazione

Il rilevamento degli oggetti utilizza metriche diverse rispetto alla classificazione delle immagini:

- **Intersection over Union (IoU)**: Misura la sovrapposizione tra bounding box previsti e "ground truth" [reali].
- **Mean Average Precision (mAP)**: Combina precisione e richiamo in tutte le classi e soglie IoU.
- **Frames Per Second (FPS)**: Misura la velocità di rilevamento, fondamentale per le applicazioni in tempo reale su dispositivi edge.

Panoramica dei Modelli di Rilevamento degli Oggetti Pre-Addestrati

Come abbiamo visto nell'introduzione, data un'immagine o un flusso video, un modello di rilevamento degli oggetti può identificare quale, di un set noto di oggetti, potrebbe essere presente e fornire informazioni sulle loro posizioni all'interno dell'immagine.

> Si possono testare alcuni modelli comuni online visitando Object Detection - MediaPipe Studio

Su Kaggle, possiamo trovare i modelli tflite pre-addestrati più comuni da usare con Raspi, ssd_mobilenet_v1 e EfficientDet. Quei modelli sono stati addestrati sul dataset COCO (Common Objects in Context), con oltre 200.000 immagini etichettate in 91 categorie. Scaricare i modelli e caricali nella cartella `./models` in Raspi.

> In alternativa, si possono trovare i modelli e le etichette COCO su GitHub.

Per la prima parte di questo laboratorio, ci concentreremo su un modello SSD-Mobilenet V1 300x300 pre-addestrato e lo confronteremo con EfficientDet-lite0 320x320, anch'esso addestrato utilizzando il set di dati COCO 2017. Entrambi i modelli sono stati convertiti in un formato TensorFlow Lite (4,2 MB per SSD Mobilenet e 4,6 MB per EfficientDet).

> SSD-Mobilenet V2 o V3 è consigliato per progetti di "transfer learning", ma una volta che il modello TFLite V1 sarà disponibile al pubblico, lo useremo per questa panoramica.

Impostazione dell'Ambiente TFLite

Dovremmo confermare i passaggi eseguiti nell'ultimo laboratorio pratico, Image Classification, come segue:

- Aggiornamento di Raspberry Pi
- Installazione delle Librerie Richieste
- Impostazione di un Ambiente Virtuale (Facoltativo ma Consigliato)

```
source ~/tflite/bin/activate
```

- Installazione di TensorFlow Lite Runtime
- Installazione di Additional Python Libraries (all'interno dell'ambiente)

Creazione di una Directory di Lavoro:

Considerando che abbiamo creato la cartella `Documents/TFLITE` nell'ultimo Lab, creiamo ora le cartelle specifiche per questo lab di rilevamento degli oggetti:

```
cd Documents/TFLITE/
mkdir OBJ_DETECT
cd OBJ_DETECT
mkdir images
mkdir models
cd models
```

Inferenza e Post-Elaborazione

Apriamo un nuovo notebook per seguire tutti i passaggi per rilevare oggetti su un'immagine: Importare le librerie necessarie:

```
import time
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import tflite_runtime.interpreter as tflite
```

Caricare il modello TFLite e allocare i tensori:

```
model_path = "./models/ssd-mobilenet-v1-tflite-default-v1.tflite"
interpreter = tflite.Interpreter(model_path=model_path)
interpreter.allocate_tensors()
```

Ottenere i tensori di input e output.

```
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Input details ci informeranno su come il modello dovrebbe essere alimentato con un'immagine. La forma di `(1, 300, 300, 3)` con un dtype di `uint8` ci dice che un'immagine non normalizzata (intervallo di valori pixel da 0 a 255) con dimensioni (300x300x3) dovrebbe essere inserita una alla volta (Batch Dimension: 1).

Gli **output details** includono non solo le etichette ("classes") e le probabilità ("scores"), ma anche la posizione relativa della finestra dei bounding box ("box") su dove si trova l'oggetto

nell'immagine e il numero di oggetti rilevati ("num_detections"). I dettagli di output ci dicono anche che il modello può rilevare un **massimo di 10 oggetti** nell'immagine.

Quindi, per l'esempio precedente, utilizzando la stessa immagine di gatto utilizzata col *Lab di Classificazione Immagine* alla ricerca dell'output, abbiamo una **probabilità del 76%** di aver trovato un oggetto con un **ID classe di 16** su un'area delimitata da un **bounding box di [0.028011084, 0.020121813, 0.9886069, 0.802299]**. Questi quattro numeri sono correlati a `ymin`, `xmin`, `ymax` e `xmax`, le coordinate del box.

Considerando che **y** va dall'alto (`ymin`) al basso (`ymax`) e **x** va da sinistra (`xmin`) a destra (`xmax`), abbiamo, di fatto, le coordinate dell'angolo superiore/sinistro e di quello inferiore/destro. Con entrambi i bordi e conoscendo la forma dell'immagine, è possibile disegnare un rettangolo attorno all'oggetto, come mostrato nella figura sottostante:

Successivamente, dovremmo scoprire cosa significa un ID di classe uguale a 16. Aprendo il file `coco_labels.txt`, come un elenco, ogni elemento ha un indice associato e ispezionando l'indice 16, otteniamo, come previsto, `cat`. La probabilità è il valore restituito dal punteggio.
Carichiamo ora alcune immagini con più oggetti su di esse per il test.

```
img_path = "./images/cat_dog.jpeg"
orig_img = Image.open(img_path)

# Display the image
plt.figure(figsize=(8, 8))
plt.imshow(orig_img)
plt.title("Original Image")
plt.show()
```

In base ai dettagli di input, pre-elaboriamo l'immagine, modificandone la forma ed espandendone le dimensioni:

```
img = orig_img.resize((input_details[0]['shape'][1],
                  input_details[0]['shape'][2]))
input_data = np.expand_dims(img, axis=0)
input_data.shape, input_data.dtype
```

La nuova forma input_data è `(1, 300, 300, 3)` con un dtype di `uint8`, che è compatibile con quanto previsto dal modello.
Utilizzando input_data, eseguiamo l'interprete, misuriamo la latenza e otteniamo l'output:

```
start_time = time.time()
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
end_time = time.time()
inference_time = (end_time - start_time) * 1000  # Convert to milliseconds
print ("Inference time: {:.1f}ms".format(inference_time))
```

Con una latenza di circa 800 ms, possiamo ottenere 4 output distinti:

```
boxes = interpreter.get_tensor(output_details[0]['index'])[0]
classes = interpreter.get_tensor(output_details[1]['index'])[0]
scores = interpreter.get_tensor(output_details[2]['index'])[0]
num_detections = int(interpreter.get_tensor(output_details[3]['index'])[0])
```

Da una rapida occhiata, possiamo vedere che il modello ha rilevato 2 oggetti con un punteggio superiore a 0,5:

```
for i in range(num_detections):
    if scores[i] > 0.5:  # Confidence threshold
        print(f"Object {i}:")
        print(f"  Bounding Box: {boxes[i]}")
        print(f"  Confidence: {scores[i]}")
        print(f"  Class: {classes[i]}")
```

```
Object 0:
  Bounding Box: [0.4125163  0.04130688 0.997076   0.42888364]
  Confidence: 0.73828125
  Class: 16.0
Object 1:
  Bounding Box: [0.20249811 0.41268167 0.99390197 0.95425284]
  Confidence: 0.69921875
  Class: 17.0
```

E possiamo anche visualizzare i risultati:

```
plt.figure(figsize=(12, 8))
plt.imshow(orig_img)
for i in range(num_detections):
    if scores[i] > 0.5:  # Adjust threshold as needed
        ymin, xmin, ymax, xmax = boxes[i]
        (left, right, top, bottom) = (xmin * orig_img.width,
                                      xmax * orig_img.width,
                                      ymin * orig_img.height,
                                      ymax * orig_img.height)
        rect = plt.Rectangle((left, top), right-left, bottom-top,
                             fill=False, color='red', linewidth=2)
        plt.gca().add_patch(rect)
        class_id = int(classes[i])
        class_name = labels[class_id]
        plt.text(left, top-10, f'{class_name}: {scores[i]:.2f}',
                 color='red', fontsize=12, backgroundcolor='white')
```

EfficientDet

EfficientDet non è tecnicamente un modello SSD (Single Shot Detector), ma condivide alcune somiglianze e si basa su idee provenienti da SSD e altre architetture di rilevamento degli oggetti:

1. EfficientDet:
    - Sviluppato dai ricercatori di Google nel 2019
    - Utilizza EfficientNet come rete backbone
    - Utilizza una nuova "bi-directional feature pyramid network (BiFPN)" [rete piramidale di feature bidirezionale]
    - Utilizza il ridimensionamento composto per ridimensionare in modo efficiente la rete backbone e i componenti di rilevamento degli oggetti.
2. Somiglianze con SSD:
    - Entrambi sono rilevatori a stadio singolo, il che significa che eseguono la localizzazione e la classificazione degli oggetti in un singolo passaggio "forward" [in avanti].
    - Entrambi utilizzano mappe di feature multiscala per rilevare oggetti a diverse scale.
3. Differenze principali:
    - Backbone: SSD in genere utilizza VGG o MobileNet, mentre EfficientDet utilizza EfficientNet.
    - Fusione di funzionalità: SSD utilizza una semplice piramide di funzionalità, mentre EfficientDet utilizza il più avanzato BiFPN.
    - Metodo di ridimensionamento: EfficientDet introduce il ridimensionamento composto per tutti i componenti della rete
4. Vantaggi di EfficientDet:
    - In genere, raggiunge migliori compromessi tra accuratezza ed efficienza rispetto a SSD e molti altri modelli di rilevamento di oggetti.
    - Un ridimensionamento più flessibile consente una famiglia di modelli con diversi compromessi tra dimensioni e prestazioni.

Sebbene EfficientDet non sia un modello SSD, può essere visto come un'evoluzione delle architetture di rilevamento a fase singola, che incorpora tecniche più avanzate per migliorare efficienza e precisione. Quando si utilizza EfficientDet, possiamo aspettarci strutture di output simili a SSD (ad esempio, bounding box e punteggi di classe).

> Su GitHub, si trova un altro notebook che esplora il modello EfficientDet che abbiamo realizzato con SSD MobileNet.

Progetto di Rilevamento di Oggetti

Ora, svilupperemo un progetto completo di classificazione delle immagini dalla raccolta dei dati all'addestramento e all'implementazione. Come abbiamo fatto con il progetto di classificazione delle immagini, il modello addestrato e convertito verrà utilizzato per l'inferenza.
Utilizzeremo lo stesso set di dati per addestrare 3 modelli: SSD-MobileNet V2, FOMO e YOLO.

L'Obiettivo

Tutti i progetti di Machine Learning devono iniziare con un obiettivo. Supponiamo di trovarci in una struttura industriale e di dover ordinare e contare **ruote** e **scatole** speciali.

In altre parole, dovremmo eseguire una classificazione multi-etichetta, in cui ogni immagine può avere tre classi:

- Background (nessun oggetto)
- Box [Scatola]
- Wheel [Ruota]

Raccolta Dati Grezzi

Una volta definito l'obiettivo del nostro progetto di apprendimento automatico, il passaggio successivo, e più cruciale, è la raccolta del dataset. Possiamo utilizzare un telefono, il Raspi o un mix per creare il set di dati grezzi (senza etichette). Usiamo la semplice app Web sul nostro Raspberry Pi per visualizzare le immagini `QVGA (320 x 240)` catturate in un browser. Da GitHub, si prende lo script Python get_img_data.py e lo si apre nel terminale:

```
python3 get_img_data.py
```

Accedere all'interfaccia web:

- Sul Raspberry Pi stesso (se si ha una GUI): si apre un browser web e si va su`http://localhost:5000`
- Da un altro dispositivo sulla stessa rete: aprire un browser web e andare su `http://<raspberry_pi_ip>:5000` (Sostituire `<raspberry_pi_ip>` con l'indirizzo IP del Raspberry Pi). Per esempio: `http://192.168.4.210:5000/`

Lo script Python crea un'interfaccia basata sul Web per catturare e organizzare set di dati di immagini utilizzando un Raspberry Pi e la sua fotocamera. È utile per progetti di

apprendimento automatico che richiedono dati di immagini etichettati o meno, come nel nostro caso.

Accedere all'interfaccia Web da un browser, inserire un'etichetta generica per le immagini da catturare e premere `Start Capture`.

Notare che le immagini da catturare avranno più etichette che dovranno essere definite in seguito.

Utilizzare l'anteprima live per posizionare la fotocamera e cliccare su `Capture Image` per salvare le immagini sotto l'etichetta corrente (in questo caso, `box-wheel`.

Quando abbiamo abbastanza immagini, possiamo premere `Stop Capture`. Le immagini catturate vengono salvate nella cartella dataset/box-wheel:

```
marcelo_rovai — mjrovai@raspi-zero: ~/Documents/TFLITE/OBJ_DETECT/dataset — ssh mjrovai@192.168.4.210 — 102×11
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT $ ls
Untitled.ipynb  dataset  get_img_data.py  images  models
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT $ cd dataset
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT/dataset $ ls
box-wheel
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT/dataset $ ls box-wheel
image_20240903-224450.jpg  image_20240903-224513.jpg  image_20240903-224530.jpg
image_20240903-224452.jpg  image_20240903-224516.jpg  image_20240903-224533.jpg
image_20240903-224458.jpg  image_20240903-224520.jpg  image_20240903-224535.jpg
image_20240903-224504.jpg  image_20240903-224524.jpg
(tflite) mjrovai@raspi-zero:~/Documents/TFLITE/OBJ_DETECT/dataset $
```

Acquisire circa 60 immagini. Provare ad acquisire con diverse angolazioni, sfondi e condizioni di luce. Filezilla può trasferire il dataset raw creato sul computer principale.

Etichettatura dei Dati

Il passaggio successivo in un progetto di Object Detect è creare un dataset etichettato. Dovremmo etichettare le immagini del dataset raw, creando riquadri di delimitazione attorno agli oggetti di ogni immagine (box e ruota). Possiamo usare strumenti di etichettatura come LabelImg, CVAT, Roboflow, o persino Edge Impulse Studio. Dopo aver esplorato lo strumento Edge Impulse in altri laboratori, usiamo Roboflow qui.

Stiamo usando Roboflow (versione gratuita) qui per due motivi principali. 1) Possiamo avere un auto-labeler e 2) Il dataset annotato è disponibile in diversi formati e può essere utilizzato sia su Edge Impulse Studio (lo useremo per MobileNet V2 e FOMO train) sia su CoLab (YOLOv8 train), ad esempio. Avendo il dataset annotato su Edge Impulse (account gratuito), non è possibile utilizzarlo per il training su altre piattaforme.

Dovremmo caricare il dataset grezzo su Roboflow. Creare un account gratuito lì e avviare un nuovo progetto, ad esempio ("box-versus-wheel").

Non entreremo nei dettagli del processo Roboflow dato che sono disponibili molti tutorial.

Annotazione

Una volta creato il progetto e caricato il dataset, si devono effettuare le annotazioni utilizzando lo strumento “Auto-Label”. Notare che si può anche caricare immagini con solo uno sfondo, che dovrebbero essere salvate senza annotazioni.

Una volta annotate tutte le immagini, si devono dividere in training, validation e testing.

Pre-elaborazione dei Dati

L'ultimo passaggio con il dataset è la pre-elaborazione per generare una versione finale per il training. Ridimensioniamo tutte le immagini a 320x320 e generiamo versioni aumentate di ogni immagine (augmentation) per creare nuovi esempi di training da cui il nostro modello può imparare.

Per l'augmentation, ruoteremo le immagini (+/-15°), ritaglieremo e varieremo la luminosità e l'esposizione.

Alla fine del processo avremo 153 immagini.

Ora si deve esportare il dataset annotato in un formato che Edge Impulse, Ultralitics e altri framework/strumenti possano comprendere, ad esempio `YOLOv8`. Scarichiamo una versione compressa del dataset sul nostro desktop.

Qui è possibile rivedere come è stato strutturato il dataset

Ci sono 3 cartelle separate, una per ogni suddivisione (`train`/`test`/`valid`). Per ognuna di esse ci sono 2 sottocartelle, `images` e `labels`. Le immagini sono archiviate come **image_id.jpg** e **images_id.txt**, dove "image_id" è univoco per ogni immagine.
Il formato del file delle etichette sarà `class_id coordinate del riquadro di delimitazione`, dove nel nostro caso class_id sarà `0` per `box` e `1` per `wheel`. L'ID numerico (o, 1, 2...) seguirà l'ordine alfabetico del nome della classe.
Il file `data.yaml` contiene informazioni sul set di dati come i nomi delle classi (`names: ['box', 'wheel']`) seguendo il formato YOLO.
E questo è tutto! Siamo pronti per iniziare l'addestramento utilizzando Edge Impulse Studio (come faremo nel passaggio successivo), Ultralytics (come faremo quando discuteremo di YOLO) o persino l'addestramento da zero su CoLab (come abbiamo fatto col dataset Cifar-10 nel laboratorio di classificazione delle immagini).

Il dataset pre-elaborato si trova sul sito Roboflow, o qui:

Addestramento di un Modello SSD MobileNet su Edge Impulse Studio

Si va su Edge Impulse Studio, si inseriscono le proprie credenziali in **Login** (o si crea un account) e si avvia un nuovo progetto.

Qui, è possibile clonare il progetto sviluppato per questo laboratorio pratico: Raspi - Object Detection.

Nella scheda `Dashboard` del progetto, si va in basso e su **Project info** e per "Labeling method" si seleziona `Bounding boxes (object detection)`

Caricamento dei dati annotati

Su Studio, si va alla scheda `Data acquisition` e nella sezione `UPLOAD DATA` si carica dal computer il set di dati non elaborato.
Possiamo usare l'opzione `Select a folder`, scegliendo, ad esempio, la cartella `train` nel computer, che contiene due sottocartelle, `images` e `labels`. Selezionare `Image label format`, "YOLO TXT", caricare nella categoria `Training` e premere `Upload data`.

Ripetere il processo per i dati di test (caricare entrambe le cartelle, test e convalida). Alla fine del processo di caricamento, si ottiene il set di dati annotato di 153 immagini suddivise in train/test (84%/16%).

Notare che le etichette saranno archiviate nei file di etichette `0` e `1`, che sono equivalenti a `box` e `wheel`.

Impulse Design

La prima cosa da definire quando entriamo nella fase `Create impulse` è descrivere il dispositivo target per la distribuzione. Apparirà una finestra pop-up. Selezioneremo Raspberry 4, un dispositivo intermedio tra Raspi-Zero e Raspi-5.

Questa scelta non interferirà con l'addestramento; ci darà solo un'idea della latenza del modello su quel target specifico.

In questa fase, si deve definire come:

- Il **pre-processing** consiste nel ridimensionare le singole immagini. Nel nostro caso, le immagini sono state pre-elaborate su Roboflow, a `320x320`, quindi teniamole. La modifica delle dimensioni non avrà importanza qui perché le immagini sono già quadrate. Se si carica un'immagine rettangolare, la si deve schiacciare (forma quadrata, senza ritagliarla). In seguito, si potrebbe definire se le immagini vengono convertite da RGB a scala di grigi o meno.
- **Design a Model**, in questo caso, "Object Detection".

Pre-elaborazione di tutti i dataset

Nella sezione `Image`, selezionare **Color depth** come `RGB` e premere `Save parameters`.

Lo Studio passa automaticamente alla sezione successiva, `Generate features`, dove tutti i campioni verranno pre-elaborati, ottenendo 480 oggetti: 207 box e 273 ruote.

L'esploratore di feature mostra che tutti i campioni evidenziano una buona separazione dopo la generazione delle feature.

Progettazione, Addestramento e Test del Modello

Per l'addestramento, dovremmo selezionare un modello pre-addestrato. Usiamo **MobileNetV2 SSD FPN-Lite (solo 320x320)** . È un modello di rilevamento oggetti pre-addestrato progettato per individuare fino a 10 oggetti all'interno di un'immagine, generando un riquadro di delimitazione per ogni oggetto rilevato. Il modello è di circa 3,7 MB. Supporta un input RGB a 320x320px.

Per quanto riguarda gli iperparametri di training, il modello verrà addestrato con:

- Epochs: 25
- Batch size: 32
- Learning Rate: 0.15.

Per la convalida durante l'addestramento, il 20% del set di dati (*validation_dataset*) verrà risparmiato.

Di conseguenza, il modello termina con un punteggio di precisione complessivo (basato su COCO mAP) dell'88,8%, superiore al risultato ottenuto utilizzando i dati di test (83,3%).

Distribuzione del modello

Abbiamo due modi per distribuire il modello:

- **Modello TFLite**, che consente di distribuire il modello addestrato come `.tflite` affinché Raspi lo esegua tramite Python.
- **Linux (AARCH64)**, un binario per Linux (AARCH64), implementa il protocollo Edge Impulse Linux, che consente di eseguire i modelli su qualsiasi scheda di sviluppo basata su Linux, ad esempio con SDK per Python. Consulta la documentazione per maggiori informazioni e le istruzioni di configurazione.

Distribuiamo il **modello TFLite**. Nella scheda `Dashboard`, si va su Transfer learning model (int8 quantized) e si clicca sull'icona di download:

Trasferire il modello dal computer alla cartella Raspi `./models` e catturare o ottenere alcune immagini per l'inferenza e salvale nella cartella `./images`.

Inferenza e Post-Elaborazione

L'inferenza può essere fatta come discusso nella *Panoramica dei Modelli di Rilevamento degli Oggetti Pre-Addestrati*. Iniziamo un nuovo notebook per seguire tutti i passaggi per rilevare cubi e ruote su un'immagine.

Importare le librerie necessarie:

```
import time
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from PIL import Image
import tflite_runtime.interpreter as tflite
```

Definire il percorso del modello e delle etichette:

```
model_path = "./models/ei-raspi-object-detection-SSD-MobileNetv2-320x0320-\
int8.lite"
labels = ['box', 'wheel']
```

Ricordare che il modello restituirà l'ID della classe come valori (0 e 1), seguendo un ordine alfabetico in base ai nomi delle classi.

Caricare il modello, allocare i tensori e ottenere i dettagli dei tensori di input e output:

```
# Load the TFLite model
interpreter = tflite.Interpreter(model_path=model_path)
interpreter.allocate_tensors()

# Get input and output tensors
input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()
```

Una differenza fondamentale da notare è che il `dtype` dei dettagli di input del modello è ora `int8`, il che significa che i valori di input vanno da -128 a +127, mentre ogni pixel della nostra immagine raw va da 0 a 256. Ciò significa che dovremmo pre-elaborare l'immagine per farla corrispondere. Possiamo controllare qui:

```
input_dtype = input_details[0]['dtype']
input_dtype
numpy.int8
```

Quindi, apriamo l'immagine e mostriamola:

```
# Load the image
img_path = "./images/box_2_wheel_2.jpg"
orig_img = Image.open(img_path)

# Display the image
plt.figure(figsize=(6, 6))
plt.imshow(orig_img)
plt.title("Original Image")
plt.show()
```

Ed eseguiamo la pre-elaborazione:

```
scale, zero_point = input_details[0]['quantization']
img = orig_img.resize((input_details[0]['shape'][1],
                  input_details[0]['shape'][2]))
img_array = np.array(img, dtype=np.float32) / 255.0
img_array = (img_array / scale + zero_point).clip(-128, 127).astype(np.int8)
input_data = np.expand_dims(img_array, axis=0)
```

Controllando i dati di input, possiamo verificare che il tensore di input è compatibile con quanto previsto dal modello:

```
input_data.shape, input_data.dtype
((1, 320, 320, 3), dtype('int8'))
```

Adesso è il momento di effettuare l'inferenza. Calcoliamo anche la latenza del modello:

```
# Inference on Raspi-Zero
start_time = time.time()
interpreter.set_tensor(input_details[0]['index'], input_data)
interpreter.invoke()
end_time = time.time()
inference_time = (end_time - start_time) * 1000  # Convert to milliseconds
print ("Inference time: {:.1f}ms".format(inference_time))
```

Il modello impiegherà circa 600 ms per eseguire l'inferenza nel Raspi-Zero, che è circa 5 volte più lungo di un Raspi-5.
Ora possiamo ottenere le classi di output degli oggetti rilevati, le coordinate dei suoi bounding box e le probabilità.

```
boxes = interpreter.get_tensor(output_details[1]['index'])[0]
classes = interpreter.get_tensor(output_details[3]['index'])[0]
scores = interpreter.get_tensor(output_details[0]['index'])[0]
num_detections = int(interpreter.get_tensor(output_details[2]['index'])[0])
for i in range(num_detections):
    if scores[i] > 0.5:  # Confidence threshold
        print(f"Object {i}:")
        print(f"  Bounding Box: {boxes[i]}")
        print(f"  Confidence: {scores[i]}")
        print(f"  Class: {classes[i]}")
```

```
Object 0:
  Bounding Box: [0.01461247 0.38439587 0.2793928  0.62159896]
  Confidence: 0.86328125
  Class: 1.0
Object 1:
  Bounding Box: [0.19234724 0.6176628  0.5012042  0.888332  ]
  Confidence: 0.86328125
  Class: 1.0
Object 2:
  Bounding Box: [0.5792029  0.19102246 0.9971932  0.47538966]
  Confidence: 0.7734375
  Class: 0.0
Object 3:
  Bounding Box: [0.5792029  0.68904555 0.9971932  0.97973716]
  Confidence: 0.6484375
  Class: 0.0
```

Dai risultati, possiamo vedere che sono stati rilevati 4 oggetti: due con ID classe 0 (`box`) e due con ID classe 1 (`wheel`), cosa è corretto!
Visualizziamo il risultato per un `threshold` [soglia] di 0,5

```
threshold = 0.5
plt.figure(figsize=(6,6))
plt.imshow(orig_img)
for i in range(num_detections):
    if scores[i] > threshold:
        ymin, xmin, ymax, xmax = boxes[i]
        (left, right, top, bottom) = (xmin * orig_img.width,
                                      xmax * orig_img.width,
                                      ymin * orig_img.height,
                                      ymax * orig_img.height)
        rect = plt.Rectangle((left, top), right-left, bottom-top,
                             fill=False, color='red', linewidth=2)
        plt.gca().add_patch(rect)
```

```
        class_id = int(classes[i])
        class_name = labels[class_id]
        plt.text(left, top-10, f'{class_name}: {scores[i]:.2f}',
                 color='red', fontsize=12, backgroundcolor='white')
```

Ma cosa succede se riduciamo la soglia a 0.3, ad esempio?

Cominciamo a vedere falsi positivi e **rilevazioni multiple**, in cui il modello rileva lo stesso oggetto più volte con diversi livelli di confidenza e bounding box leggermente diversi.
Di solito, a volte, dobbiamo regolare la soglia su valori più piccoli per catturare tutti gli oggetti, evitando falsi negativi, che porterebbero a rilevazioni multiple.

Per migliorare i risultati di rilevamento, dovremmo implementare **Non-Maximum Suppression (NMS)**, che aiuta a eliminare le bounding box sovrapposte e mantiene solo il rilevamento più affidabile.
Per questo, creiamo una funzione generale denominata `non_max_suppression()`, con il ruolo di perfezionare i risultati di rilevamento degli oggetti eliminando le bounding box ridondanti e sovrapposte. Ciò è possibile selezionando in modo iterativo la rilevazione con il punteggio di confidenza più elevato e rimuovendo altre rilevazioni significativamente sovrapposte in base a una soglia di "Intersection over Union (IoU)" [intersezione su unione].

```
def non_max_suppression(boxes, scores, threshold):
    # Convert to corner coordinates
    x1 = boxes[:, 0]
    y1 = boxes[:, 1]
    x2 = boxes[:, 2]
    y2 = boxes[:, 3]

    areas = (x2 - x1 + 1) * (y2 - y1 + 1)
    order = scores.argsort()[::-1]

    keep = []
    while order.size > 0:
        i = order[0]
        keep.append(i)
        xx1 = np.maximum(x1[i], x1[order[1:]])
        yy1 = np.maximum(y1[i], y1[order[1:]])
        xx2 = np.minimum(x2[i], x2[order[1:]])
        yy2 = np.minimum(y2[i], y2[order[1:]])

        w = np.maximum(0.0, xx2 - xx1 + 1)
        h = np.maximum(0.0, yy2 - yy1 + 1)
        inter = w * h
        ovr = inter / (areas[i] + areas[order[1:]] - inter)

        inds = np.where(ovr <= threshold)[0]
        order = order[inds + 1]

    return keep
```

Come funziona:

1. Sorting: Inizia ordinando tutte le rilevazioni in base ai loro punteggi di confidenza, dal più alto al più basso.
2. Selection: Seleziona la casella con il punteggio più alto e la aggiunge all'elenco finale delle rilevazioni.
3. Comparison: Questa casella selezionata viene confrontata con tutte le restanti caselle con punteggio più basso.
4. Elimination: Qualsiasi casella che si sovrapponga in modo significativo (oltre la soglia IoU) alla casella selezionata viene eliminata.
5. Iteration: Questo processo si ripete con la casella con il punteggio più alto successivo finché tutte le caselle non vengono elaborate.

Ora, possiamo definire una funzione di visualizzazione più precisa che prenderà in considerazione una soglia IoU, rilevando solo gli oggetti selezionati dalla funzione `non_max_suppression`:

```
def visualize_detections(image, boxes, classes, scores,
                         labels, threshold, iou_threshold):
    if isinstance(image, Image.Image):
        image_np = np.array(image)
    else:
        image_np = image

    height, width = image_np.shape[:2]

    # Convert normalized coordinates to pixel coordinates
    boxes_pixel = boxes * np.array([height, width, height, width])
```

```
    # Apply NMS
    keep = non_max_suppression(boxes_pixel, scores, iou_threshold)

    # Set the figure size to 12x8 inches
    fig, ax = plt.subplots(1, figsize=(12, 8))

    ax.imshow(image_np)

    for i in keep:
        if scores[i] > threshold:
            ymin, xmin, ymax, xmax = boxes[i]
            rect = patches.Rectangle((xmin * width, ymin * height),
                                     (xmax - xmin) * width,
                                     (ymax - ymin) * height,
                                     linewidth=2, edgecolor='r', facecolor='none')
            ax.add_patch(rect)
            class_name = labels[int(classes[i])]
            ax.text(xmin * width, ymin * height - 10,
                    f'{class_name}: {scores[i]:.2f}', color='red',
                    fontsize=12, backgroundcolor='white')

    plt.show()
```

Ora possiamo creare una funzione che chiamerà le altre, eseguendo l'inferenza su qualsiasi immagine:

```
def detect_objects(img_path, conf=0.5, iou=0.5):
    orig_img = Image.open(img_path)
    scale, zero_point = input_details[0]['quantization']
    img = orig_img.resize((input_details[0]['shape'][1],
                      input_details[0]['shape'][2]))
    img_array = np.array(img, dtype=np.float32) / 255.0
    img_array = (img_array / scale + zero_point).clip(-128, 127).\
    astype(np.int8)
    input_data = np.expand_dims(img_array, axis=0)

    # Inference on Raspi-Zero
    start_time = time.time()
    interpreter.set_tensor(input_details[0]['index'], input_data)
    interpreter.invoke()
    end_time = time.time()
    inference_time = (end_time - start_time) * 1000  # Convert to ms
    print ("Inference time: {:.1f}ms".format(inference_time))

    # Extract the outputs
    boxes = interpreter.get_tensor(output_details[1]['index'])[0]
    classes = interpreter.get_tensor(output_details[3]['index'])[0]
    scores = interpreter.get_tensor(output_details[0]['index'])[0]
    num_detections = int(interpreter.get_tensor(output_details[2]['index'])[0])

    visualize_detections(orig_img, boxes, classes, scores, labels,
                         threshold=conf,
                         iou_threshold=iou)
```

Ora, eseguendo il codice, ottenendo di nuovo la stessa immagine con una soglia di confidenza di 0.3, ma con un piccolo IoU:

```
img_path = "./images/box_2_wheel_2.jpg"
detect_objects(img_path, conf=0.3,iou=0.05)
```

Training di un modello FOMO su Edge Impulse Studio

L'inferenza con il modello SSD MobileNet ha funzionato bene, ma la latenza era significativamente alta. L'inferenza variava da 0.5 a 1.3 secondi su un Raspi-Zero, il che significa circa o meno di 1 FPS (1 frame al secondo). Un'alternativa per accelerare il processo è usare FOMO (Faster Objects, More Objects).
Questo nuovo algoritmo di apprendimento automatico consente di contare più oggetti e di trovare la loro posizione in un'immagine in tempo reale utilizzando fino a 30 volte meno potenza di elaborazione e memoria rispetto a MobileNet SSD o YOLO. Il motivo principale per cui ciò è possibile è che mentre altri modelli calcolano le dimensioni dell'oggetto disegnando un quadrato attorno ad esso (bounding box), FOMO ignora le dimensioni dell'immagine, fornendo solo le informazioni sulla posizione dell'oggetto nell'immagine tramite le sue coordinate del centroide.

Come funziona FOMO?

In una tipica pipeline di rilevamento degli oggetti, la prima fase consiste nell'estrarre le feature dall'immagine di input. **FOMO sfrutta MobileNetV2 per eseguire questa attività**. MobileNetV2 elabora l'immagine di input per produrre una feature map che cattura le caratteristiche essenziali, come texture, forme e bordi degli oggetti, in modo computazionalmente efficiente.

Una volta estratte queste feature, l'architettura più semplice di FOMO, focalizzata sul rilevamento del punto centrale, interpreta la feature map per determinare dove si trovano gli oggetti nell'immagine. L'output è una griglia di celle, in cui ogni cella rappresenta se è stato rilevato o meno un centro dell'oggetto. Il modello restituisce uno o più punteggi di confidenza per ogni cella, indicando la probabilità che un oggetto sia presente.
Vediamo come funziona su un'immagine.
FOMO divide l'immagine in blocchi di pixel usando un fattore di 8. Per l'input di 96x96, la griglia è 12x12 (96/8=12). Per un 160x160, la griglia sarà 20x20 e così via. Successivamente, FOMO eseguirà un classificatore attraverso ogni blocco di pixel per calcolare la probabilità che ci sia un box o una ruota in ognuno di essi e, successivamente, determinerà le regioni che hanno la più alta probabilità di contenere l'oggetto (se un blocco di pixel non ha oggetti, verrà classificato come *background*). Dalla sovrapposizione della regione finale, FOMO fornisce le coordinate (relative alle dimensioni dell'immagine) del centroide di questa regione.

Compromesso Tra Velocità e Precisione:

- **Risoluzione della Griglia**: FOMO utilizza una griglia di risoluzione fissa, il che significa che ogni cella può rilevare se un oggetto è presente in quella parte dell'immagine. Sebbene non fornisca un'elevata precisione di localizzazione, fa un compromesso essendo veloce e computazionalmente leggero, il che è fondamentale per i dispositivi edge.
- **Rilevamento Multi-Oggetto**: Poiché ogni cella è indipendente, FOMO può rilevare più oggetti contemporaneamente in un'immagine identificando più centri.

Impulse Design, nuovo Training e Test

Tornare a Edge Impulse Studio e nella scheda `Experiments`, creare un altro impulso. Ora, le immagini di input dovrebbero essere 160x160 (questa è la dimensione di input prevista per MobilenetV2).

Nella scheda `Image`, generare le feature e spostarsi nella scheda `Object detection`.

Dovremmo selezionare un modello pre-addestrato per l'addestramento. Usiamo **FOMO (Faster Objects, More Objects) MobileNetV2 0.35.**

Per quanto riguarda gli iperparametri di training, il modello verrà addestrato con:

- Epochs: 30
- Batch size: 32
- Learning Rate: 0.001.

Per la convalida durante l'addestramento, il 20% del set di dati (*validation_dataset*) verrà risparmiato. Non applicheremo il "Data Augmentation" per il restante 80% (*train_dataset*) perché il dataset è già stato aumentato durante la fase di etichettatura su Roboflow.
Di conseguenza, il modello termina con un punteggio F1 complessivo del 93,3% con una latenza impressionante di 8 ms (Raspi-4), circa 60 volte inferiore a quella ottenuta con SSD MovileNetV2.

Notare che FOMO ha aggiunto automaticamente una terza etichetta di background ai due *box* (0) e *ruote* (1) definite in precedenza.

Nella scheda `Model testing`, possiamo vedere che l'accuratezza era del 94%. Ecco uno dei risultati del campione di test:

Nelle attività di rilevamento di oggetti, l'accuratezza non è in genere la metrica di valutazione primaria. Il rilevamento di oggetti comporta la classificazione degli oggetti e la definizione di riquadri di delimitazione attorno a essi, il che lo rende un problema più complesso della semplice classificazione. Il problema è che non abbiamo il riquadro di delimitazione, solo i centroidi. In breve, usare l'accuratezza come metrica potrebbe essere fuorviante e potrebbe non fornire una comprensione completa delle prestazioni del modello.

Distribuzione del modello

Come abbiamo fatto nella sezione precedente, possiamo distribuire il modello addestrato come TFLite o Linux (AARCH64). Ora facciamolo come **Linux (AARCH64)**, un binario che implementa il protocollo Edge Impulse Linux.
Edge Impulse per i modelli Linux viene fornito in formato `.eim`. Questo eseguibile contiene il nostro "impulso completo" creato in Edge Impulse Studio. L'impulso è costituito dai blocchi di elaborazione del segnale e da qualsiasi blocco di apprendimento e anomalia che abbiamo aggiunto e addestrato. È compilato con ottimizzazioni per il nostro processore o GPU (ad esempio, istruzioni NEON su core ARM), più un semplice layer IPC (su un socket Unix).
Nella scheda `Deploy`, selezionare l'opzione `Linux (AARCH64)`, il modello `int8` e premere `Build`.

Il modello verrà scaricato automaticamente sul computer.
Sul nostro Raspi, creiamo una nuova area di lavoro:

```
cd ~
cd Documents
mkdir EI_Linux
cd EI_Linux
mkdir models
mkdir images
```

Rinominare il modello per facilitarne l'identificazione:
Ad esempio, `raspi-object-detection-linux-aarch64-FOMO-int8.eim` e trasferirlo nella nuova cartella Raspi `./models` e catturare o ottenere alcune immagini per l'inferenza e salvarle nella cartella `./images`.

Inferenza e Post-Elaborazione

L'inferenza verrà effettuata utilizzando Linux Python SDK. Questa libreria consente di eseguire modelli di apprendimento automatico e raccogliere dati dei sensori su macchine Linux utilizzando Python. L'SDK è open source e ospitato su GitHub: edgeimpulse/linux-sdk-python.

Impostiamo un ambiente virtuale per lavorare con Linux Python SDK

```
python3 -m venv ~/eilinux
source ~/eilinux/bin/activate
```

E installare tutte le librerie necessarie:

```
sudo apt-get update
sudo apt-get install libatlas-base-dev libportaudio0 libportaudio2
sudo apt-get installlibportaudiocpp0 portaudio19-dev

pip3 install edge_impulse_linux -i https://pypi.python.org/simple
pip3 install Pillow matplotlib pyaudio opencv-contrib-python

sudo apt-get install portaudio19-dev
pip3 install pyaudio
pip3 install opencv-contrib-python
```

Consentire al modello di essere eseguibile.

```
chmod +x raspi-object-detection-linux-aarch64-FOMO-int8.eim
```

Installare Jupiter Notebook nel nuovo ambiente

```
pip3 install jupyter
```

Eseguire un notebook in locale (su Raspi-4 o 5 con desktop)

```
jupyter notebook
```

o sul browser del computer:

```
jupyter notebook --ip=192.168.4.210 --no-browser
```

Avviamo un nuovo notebook seguendo tutti i passaggi per rilevare cubi e ruote su un'immagine utilizzando il modello FOMO e l'SDK Python di Edge Impulse Linux.
Importare le librerie necessarie:

```
import sys, time
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from PIL import Image
import cv2
from edge_impulse_linux.image import ImageImpulseRunner
```

Definire il percorso del modello e delle etichette:

```
model_file = "raspi-object-detection-linux-aarch64-int8.eim"
model_path = "models/"+ model_file # Trained ML model from Edge Impulse
labels = ['box', 'wheel']
```

> Ricordare che il modello restituirà l'ID della classe come valori (0 e 1), seguendo un ordine alfabetico in base ai nomi delle classi.

Caricare e inizializzare il modello:

```
# Load the model file
runner = ImageImpulseRunner(model_path)
```

```
# Initialize model
model_info = runner.init()
```

Il `model_info` conterrà informazioni critiche sul modello. Tuttavia, a differenza dell'interprete TFLite, la libreria EI Linux Python SDK preparerà ora il modello per l'inferenza.
Quindi, apriamo l'immagine e mostriamola (ora, per compatibilità, useremo OpenCV, la libreria CV utilizzata internamente da EI. OpenCV legge l'immagine come BGR, quindi dovremo convertirla in RGB:

```
# Load the image
img_path = "./images/1_box_1_wheel.jpg"
orig_img = cv2.imread(img_path)
img_rgb = cv2.cvtColor(orig_img, cv2.COLOR_BGR2RGB)

# Display the image
plt.imshow(img_rgb)
plt.title("Original Image")
plt.show()
```

Ora otterremo le feature e l'immagine preelaborata (`ritagliata`) utilizzando `runner`:

```
features, cropped = runner.get_features_from_image_auto_studio_setings(img_rgb)
```

Ed eseguiamo l'inferenza. Calcoliamo anche la latenza del modello:

```
res = runner.classify(features)
```

Otteniamo le classi di output degli oggetti rilevati, i loro centroidi dei bounding box e le probabilità.

```
print('Found %d bounding boxes (%d ms.)' % (
  len(res["result"]["bounding_boxes"]),
  res['timing']['dsp'] + res['timing']['classification']))
for bb in res["result"]["bounding_boxes"]:
    print('\t%s (%.2f): x=%d y=%d w=%d h=%d' % (
      bb['label'], bb['value'], bb['x'],
      bb['y'], bb['width'], bb['height']))
Found 2 bounding boxes (29 ms.)
    1 (0.91): x=112 y=40 w=16 h=16
    0 (0.75): x=48 y=56 w=8 h=8
```

I risultati mostrano che sono stati rilevati due oggetti: uno con ID classe 0 (`box`) e uno con ID classe 1 (`wheel`), il che è corretto!
Visualizziamo il risultato (Il `threshold` [soglia] è 0.5, il valore predefinito impostato durante il test del modello su Edge Impulse Studio).

```
print('\tFound %d bounding boxes (latency: %d ms)' % (
  len(res["result"]["bounding_boxes"]),
  res['timing']['dsp'] + res['timing']['classification']))
plt.figure(figsize=(5,5))
plt.imshow(cropped)

# Go through each of the returned bounding boxes
bboxes = res['result']['bounding_boxes']
for bbox in bboxes:

    # Get the corners of the bounding box
    left = bbox['x']
    top = bbox['y']
    width = bbox['width']
    height = bbox['height']

    # Draw a circle centered on the detection
    circ = plt.Circle((left+width//2, top+height//2), 5,
                      fill=False, color='red', linewidth=3)
    plt.gca().add_patch(circ)
    class_id = int(bbox['label'])
    class_name = labels[class_id]
    plt.text(left, top-10, f'{class_name}: {bbox["value"]:.2f}',
             color='red', fontsize=12, backgroundcolor='white')
plt.show()
```

Esplorazione di un Modello YOLO tramite Ultralitics

Per questo laboratorio, esploreremo YOLOv8. Ultralytics YOLOv8 è una versione dell'acclamato modello di rilevamento di oggetti in tempo reale e segmentazione delle

immagini, YOLO. YOLOv8 è basato su progressi all'avanguardia nel deep learning e nella visione artificiale, offrendo prestazioni senza pari in termini di velocità e precisione. Il suo design semplificato lo rende adatto a varie applicazioni e facilmente adattabile a diverse piattaforme hardware, dai dispositivi edge alle API cloud.

A proposito del modello YOLO

Il modello YOLO (You Only Look Once) è un algoritmo di rilevamento di oggetti altamente efficiente e ampiamente utilizzato, noto per le sue capacità di elaborazione in tempo reale. A differenza dei tradizionali sistemi di rilevamento di oggetti che riutilizzano classificatori o localizzatori per eseguire il rilevamento, YOLO inquadra il problema del rilevamento come un singolo compito di regressione. Questo approccio innovativo consente a YOLO di prevedere simultaneamente più bounding box e le relative probabilità di classe da immagini complete in un'unica valutazione, aumentando notevolmente la sua velocità.

Caratteristiche Principali:

1. **Architettura "ingle Network"**:
    - YOLO impiega una singola rete neurale per elaborare l'intera immagine. Questa rete divide l'immagine in una griglia e, per ogni cella della griglia, prevede direttamente i bounding box e le relative probabilità di classe. Questa formazione end-to-end migliora la velocità e semplifica l'architettura del modello.
2. **Elaborazione in Tempo Reale**:
    - Una delle caratteristiche distintive di YOLO è la sua capacità di eseguire il rilevamento di oggetti in tempo reale. A seconda della versione e dell'hardware, YOLO può elaborare immagini con frame al secondo (FPS) elevati. Ciò lo rende ideale per applicazioni che richiedono un rilevamento di oggetti rapido e accurato, come videosorveglianza, guida autonoma e analisi di eventi sportivi in diretta.
3. **Evoluzione delle Versioni**:
    - Nel corso degli anni, YOLO ha subito miglioramenti significativi, da YOLOv1 all'ultimo YOLOv10. Ogni iterazione ha introdotto miglioramenti in termini di accuratezza, velocità ed efficienza. YOLOv8, ad esempio, incorpora progressi nell'architettura di rete, metodologie di training migliorate e un supporto migliore per vari hardware, garantendo prestazioni più robuste.
    - Sebbene YOLOv10 sia il membro più recente della famiglia con prestazioni incoraggianti in base alla sua documentazione, è stato appena rilasciato (maggio 2024) e non è completamente integrato con la libreria Ultralitycs. Al contrario, l'analisi della curva di precisione-richiamo suggerisce che YOLOv8 generalmente supera YOLOv9, catturando una percentuale maggiore di veri positivi e riducendo al minimo i falsi positivi in modo più efficace (per maggiori dettagli, vedere questo articolo). Quindi, questo laboratorio si basa su YOLOv8n.

4. **Precisione ed Efficienza**:
   - Mentre le prime versioni di YOLO barattavano un po' di precisione per la velocità, le versioni recenti hanno fatto notevoli progressi nell'equilibrare entrambi. I modelli più recenti sono più veloci e precisi, rilevano piccoli oggetti (come le api) e funzionano bene su set di dati complessi.
5. **Ampia Gamma di Applicazioni**:
   - La versatilità di YOLO ha portato alla sua adozione in numerosi campi. Viene utilizzato nei sistemi di monitoraggio del traffico per rilevare e contare i veicoli, nelle applicazioni di sicurezza per identificare potenziali minacce e nella tecnologia agricola per monitorare raccolti e bestiame. La sua applicazione si estende a qualsiasi dominio che richieda un rilevamento efficiente e accurato degli oggetti.
6. **Comunità e Sviluppo**:
   - YOLO continua a evolversi ed è supportato da una solida comunità di sviluppatori e ricercatori (essendo YOLOv8 molto forte). Le implementazioni open source e l'ampia documentazione lo hanno reso accessibile per la personalizzazione e l'integrazione in vari progetti. Framework di deep learning popolari come Darknet, TensorFlow e PyTorch supportano YOLO, ampliandone ulteriormente l'applicabilità.
   - Ultralytics YOLOv8 non solo può Detect [rilevare] (il nostro caso qui) ma anche Segmentare e mettere in posa con Pose modelli pre-addestrati sul set di dati COCO e YOLOv8 Classifica modelli pre-addestrati sul set di dati ImageNet. La modalità Track è disponibile per tutti i modelli Detect, Segment e Pose.

Attività supportate da Ultralytics YOLO

Installazione

Sul nostro Raspi, disattiviamo l'ambiente corrente per creare una nuova area di lavoro:

```
deactivate
cd ~
cd Documents/
mkdir YOLO
cd YOLO
mkdir models
mkdir images
```

Impostiamo un Virtual Environment per lavorare con Ultralytics YOLOv8

```
python3 -m venv ~/yolo
source ~/yolo/bin/activate
```

E installiamo i pacchetti Ultralytics per l'inferenza locale sul Raspi

1. Aggiorniamo l'elenco dei pacchetti, installiamo pip ed eseguiamo l'aggiornamento all'ultima versione:

```
sudo apt update
sudo apt install python3-pip -y
pip install -U pip
```

2. Installiamo il pacchetto pip `ultralytics` con le dipendenze opzionali:

```
pip install ultralytics[export]
```

3. Riavviamo il dispositivo:

```
sudo reboot
```

Test di YOLO

Dopo l'avvio di Raspi-Zero, attiviamo l'ambiente `yolo`, andiamo alla directory di lavoro,

```
source ~/yolo/bin/activate
cd /Documents/YOLO
```

ed eseguiamo l'inferenza su un'immagine che verrà scaricata dal sito web di Ultralytics, che utilizza il modello YOLOV8n (il più piccolo della famiglia) nel Terminal (CLI):

```
yolo predict model='yolov8n' source='https://ultralytics.com/images/bus.jpg'
```

La famiglia di modelli YOLO è pre-addestrata con il dataset COCO.

Il risultato dell'inferenza apparirà nel terminale. Nell'immagine (bus.jpg), sono state rilevate 4 `persone`, 1 `autobus` e 1 `segnale di stop`:

Inoltre, abbiamo ricevuto un messaggio che indica `Results saved to runs/detect/predict`. Ispezionando quella directory, possiamo vedere una nuova immagine salvata (bus.jpg). Scarichiamola dal Raspi-Zero sul desktop per l'ispezione:

Quindi, Ultrayitics YOLO è installato correttamente sul nostro Raspi. Ma, su Raspi-Zero, un problema è l'elevata latenza per questa inferenza, circa 18 secondi, anche con il modello più in miniatura della famiglia (YOLOv8n).

Esportazione del Modello in formato NCNN

L'implementazione di modelli di visione artificiale su dispositivi edge con potenza di calcolo limitata, come Raspi-Zero, può causare problemi di latenza. Un'alternativa è quella di utilizzare un formato ottimizzato per prestazioni ottimali. Ciò garantisce che anche i dispositivi con potenza di elaborazione limitata possano gestire bene attività di visione artificiale avanzate.

Di tutti i formati di esportazione del modello supportati da Ultralytics, NCNN è un framework di elaborazione inferenziale di reti neurali ad alte prestazioni ottimizzato per piattaforme mobili. Fin dall'inizio della progettazione, NCNN è stato profondamente attento all'implementazione e all'uso su telefoni cellulari e non aveva dipendenze di terze parti. È multipiattaforma e funziona più velocemente di tutti i framework open source noti (come TFLite).

NCNN offre le migliori prestazioni di inferenza quando si lavora con dispositivi Raspberry Pi. NCNN è altamente ottimizzato per piattaforme mobili embedded (come l'architettura ARM).

Quindi, convertiamo il nostro modello e rieseguiamo l'inferenza:

1. Esportiamo un modello PyTorch YOLOv8n in formato NCNN, creando: '/yolov8n_ncnn_model'

```
yolo export model=yolov8n.pt format=ncnn
```

2. Eseguiamo l'inferenza con il modello esportato (ora la sorgente potrebbe essere l'immagine bus.jpg scaricata dal sito Web nella directory corrente nell'ultima inferenza):

```
yolo predict model='./yolov8n_ncnn_model' source='bus.jpg'
```

La prima inferenza, quando il modello viene caricato, di solito ha una latenza elevata (circa 17 s), ma dalla seconda, è possibile notare che l'inferenza scende a circa 2 s.

Esplorare YOLO con Python

Per iniziare, chiamiamo l'interprete Python in modo da poter esplorare il funzionamento del modello YOLO, riga per riga:

```
python3
```

Ora, dovremmo chiamare la libreria YOLO da Ultralitics e caricare il modello:

```
from ultralytics import YOLO
model = YOLO('yolov8n_ncnn_model')
```

Quindi, eseguire l'inferenza su un'immagine (usiamo di nuovo `bus.jpg`):

```
img = 'bus.jpg'
result = model.predict(img, save=True, imgsz=640, conf=0.5, iou=0.3)
```

Possiamo verificare che il risultato è quasi identico a quello che otteniamo eseguendo l'inferenza a livello di terminale (CLI), tranne per il fatto che la fermata dell'autobus non è stata rilevata con il modello NCNN ridotto. Notare che la latenza è stata ridotta.
Analizziamo il contenuto di "result".
Ad esempio, possiamo vedere `result[0].boxes.data`, che mostra il risultato principale dell'inferenza, che è un profilo tensoriale (4, 6). Ogni riga è uno degli oggetti rilevati, ovvero le prime 4 colonne, le coordinate delle bounding box, la quinta, la confidenza e la sesta, la classe (in questo caso, `0: person` e `5: bus`):

Possiamo accedere a diversi risultati di inferenza separatamente, come il tempo di inferenza, e stamparli in un formato migliore:

```
inference_time = int(result[0].speed['inference'])
print(f"Inference Time: {inference_time} ms")
```

Oppure possiamo avere il numero totale di oggetti rilevati:

```
print(f'Number of objects: {len (result[0].boxes.cls)}')
```

Con Python, possiamo creare un output dettagliato che soddisfi le nostre esigenze (vedere Model Prediction with Ultralytics YOLO per maggiori dettagli). Eseguiamo uno script Python invece di inserirlo manualmente riga per riga nell'interprete, come mostrato di seguito. Usiamo `nano` come editor di testo. Per prima cosa, dovremmo creare uno script Python vuoto denominato, ad esempio, `yolov8_tests.py`:

```
nano yolov8_tests.py
```

Si inseriscono le righe di codice:

```
from ultralytics import YOLO

# Load the YOLOv8 model
model = YOLO('yolov8n_ncnn_model')

# Run inference
img = 'bus.jpg'
result = model.predict(img, save=False, imgsz=640, conf=0.5, iou=0.3)

# print the results
inference_time = int(result[0].speed['inference'])
print(f"Inference Time: {inference_time} ms")
print(f'Number of objects: {len (result[0].boxes.cls)}')
```

E si inseriscono i comandi: `[CTRL+O]` + `[INVIO]` + `[CTRL+X]` per salvare lo script Python.

Eseguire lo script:

```
python yolov8_tests.py
```

Il risultato è lo stesso dell'esecuzione dell'inferenza a livello di terminale (CLI) e con l'interprete Python nativo.

> Chiamare la libreria YOLO e caricare il modello per l'inferenza per la prima volta richiede molto tempo, ma le inferenze successive saranno molto più veloci. Ad esempio, la prima singola inferenza può richiedere diversi secondi, ma in seguito il tempo di inferenza dovrebbe essere ridotto a meno di 1 secondo.

Addestramento di YOLOv8 su un Dataset Personalizzato

Torniamo al nostro set di dati "Boxe versus Wheel", etichettato su Roboflow. Nell'opzione `Download Dataset`, invece dell'opzione `Download a zip to computer` eseguita per l'addestramento su Edge Impulse Studio, opteremo per `Show download code`. Questa opzione aprirà una finestra pop-up con un frammento di codice che dovrebbe essere incollato nel nostro notebook di training.

Per l'addestramento, adattiamo uno degli esempi pubblici disponibili da Ultralitytics ed eseguiamolo su Google Colab. Di seguito, si può trovare il mio da adattare al progetto:

- YOLOv8 Box versus Wheel Dataset Training [Open In Colab]

Punti critici sul Notebook:

1. Eseguirlo con GPU (NVidia T4 è gratuito)

2. Installare Ultralytics tramite PIP.

```
[3]
# Pip install method (recommended)

!pip install ultralytics

from IPython import display
display.clear_output()

import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.2.91 🚀 Python-3.10.12 torch-2.4.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 32.8/112.6 GB disk)
```

3. Ora, si può importare YOLO e caricare il dataset sul CoLab, incollando il codice di download che riceviamo da Roboflow. Notare che il dataset verrà montato in `/content/datasets/`:

4. È essenziale verificare e modificare il file `data.yaml` con il path corretto per le immagini (copiare il percorso su ogni cartella `images`).

```
names:
- box
- wheel
nc: 2
roboflow:
  license: CC BY 4.0
  project: box-versus-wheel-auto-dataset
  url: https://universe.roboflow.com/marcelo-rovai-riila/box-versus-wheel-auto-dataset/da
taset/5
  version: 5
  workspace: marcelo-rovai-riila
test: /content/datasets/Box-versus-Wheel-auto-dataset-5/test/images
train: /content/datasets/Box-versus-Wheel-auto-dataset-5/train/images
val: /content/datasets/Box-versus-Wheel-auto-dataset-5/valid/images
```

5. Definire i principali iperparametri da modificare rispetto ai valori di default, ad esempio:

```
MODEL = 'yolov8n.pt'
IMG_SIZE = 640
EPOCHS = 25 # For a final project, you should consider at least 100 epochs
```

6. Eseguire il training (utilizzando la CLI):

```
!yolo task=detect mode=train model={MODEL} data={dataset.location}/data.yaml epoch
s={EPOCHS} imgsz={IMG_SIZE} plots=True
```

```
25 epochs completed in 0.026 hours.
Optimizer stripped from runs/detect/train/weights/last.pt, 6.2MB
Optimizer stripped from runs/detect/train/weights/best.pt, 6.2MB

Validating runs/detect/train/weights/best.pt...
Ultralytics YOLOv8.2.91 🚀 Python-3.10.12 torch-2.4.0+cu121 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 3,006,038 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100% 1/1 [00:00<00:00,  7.61it/s]
                   all         12         65      0.997          1      0.995      0.899
                   box         11         29      0.999          1      0.995      0.903
                 wheel         11         36      0.995          1      0.995      0.896
Speed: 0.2ms preprocess, 2.6ms inference, 0.0ms loss, 3.2ms postprocess per image
```

image-20240910111319804

L'addestramento del modello ha richiesto alcuni minuti e ha prodotto risultati eccellenti (mAP50 pari a 0,995). Al termine del training, tutti i risultati vengono salvati nella cartella elencata, ad esempio: `/runs/detect/train/`. Lì si trova, ad esempio, la matrice di confusione.

7. Notare che il modello addestrato (`best.pt`) viene salvato nella cartella `/runs/detect/train/weights/`. Ora, si deve convalidare il modello addestrato con `valid/images`.

```
!yolo task=detect mode=val model={HOME}/runs/detect/train/weights/best.pt data={dataset.l
ocation}/data.yaml
I risultati erano simili al training.
```

8. Ora, si deve eseguire l'inferenza sulle immagini lasciate a parte per il test

```
!yolo task=detect mode=predict model={HOME}/runs/detect/train/weights/best.pt conf=0.25 s
ource={dataset.location}/test/images save=True
```

I risultati dell'inferenza vengono salvati nella cartella `runs/detect/predict`. Vediamone alcuni:

9. Si consiglia di esportare i risultati di train, validation e test per un Drive su Google. Per farlo, si deve montare l'unità.

```
from google.colab import drive
drive.mount('/content/gdrive')
```

e copiare il contenuto della cartella `/runs` in una cartella che si deve creare nel Drive, ad esempio:

```
!scp -r /content/runs '/content/gdrive/MyDrive/10_UNIFEI/Box_vs_Wheel_Project'
```

Inferenza col modello addestrato, usando Raspi

Scaricare il modello addestrato `/runs/detect/train/weights/best.pt` nel computer. Utilizzando l'FTP FileZilla, trasferiamo `best.pt` nella cartella dei modelli Raspi (prima del trasferimento, si può cambiare il nome del modello, ad esempio, `box_wheel_320_yolo.pt`). Con FileZilla FTP, trasferiamo alcune immagini dal set di dati di prova a `.\YOLO\images`: Torniamo alla cartella YOLO e utilizziamo l'interprete Python:

```
cd ..
python
```

Come prima, importeremo la libreria YOLO e definiremo il nostro modello convertito per rilevare le api:

```
from ultralytics import YOLO
model = YOLO('./models/box_wheel_320_yolo.pt')
```

Ora, definiamo un'immagine e chiamiamo l'inferenza (stavolta salveremo il risultato dell'immagine per la verifica esterna):

```
img = './images/1_box_1_wheel.jpg'
result = model.predict(img, save=True, imgsz=320, conf=0.5, iou=0.3)
```

Ripetiamo per diverse immagini. Il risultato dell'inferenza viene salvato sulla variabile `result` e l'immagine elaborata su `runs/detect/predict8`

```
marcelo_rovai — mjrovai@raspi-zero: ~/Documents/YOLO — ssh mjrovai@192.168.4.210 — 106×26
>>> model = YOLO('./models/box_wheel_320_yolo.pt')
>>> img = './images/1_box_1_wheel.jpg'
>>> result = model.predict(img, save=True, imgsz=320, conf=0.5, iou=0.3)

image 1/1 /home/mjrovai/Documents/YOLO/images/1_box_1_wheel.jpg: 256x320 1 box, 1 wheel, 2390.8ms
Speed: 164.3ms preprocess, 2390.8ms inference, 75.9ms postprocess per image at shape (1, 3, 256, 320)
Results saved to runs/detect/predict8
>>> img = './images/box_2_wheel_1.jpg'
>>> result = model.predict(img, save=True, imgsz=320, conf=0.5, iou=0.3)

image 1/1 /home/mjrovai/Documents/YOLO/images/box_2_wheel_1.jpg: 256x320 2 boxs, 1 wheel, 1620.4ms
Speed: 292.5ms preprocess, 1620.4ms inference, 49.2ms postprocess per image at shape (1, 3, 256, 320)
Results saved to runs/detect/predict8
>>> img = './images/2_wheel.jpg'
>>> result = model.predict(img, save=True, imgsz=320, conf=0.5, iou=0.3)

image 1/1 /home/mjrovai/Documents/YOLO/images/2_wheel.jpg: 256x320 2 wheels, 1010.3ms
Speed: 6.9ms preprocess, 1010.3ms inference, 7.7ms postprocess per image at shape (1, 3, 256, 320)
Results saved to runs/detect/predict8
>>> img = './images/1_wheel.jpg'
>>> result = model.predict(img, save=True, imgsz=320, conf=0.5, iou=0.3)

image 1/1 /home/mjrovai/Documents/YOLO/images/1_wheel.jpg: 256x320 1 wheel, 1085.1ms
Speed: 11.7ms preprocess, 1085.1ms inference, 7.4ms postprocess per image at shape (1, 3, 256, 320)
Results saved to runs/detect/predict8
>>>
```

Con FileZilla FTP, possiamo inviare il risultato dell'inferenza al Desktop per la verifica:

Possiamo vedere che il risultato dell'inferenza è eccellente! Il modello è stato addestrato in base al modello base più piccolo della famiglia YOLOv8 (YOLOv8n). Il problema è la latenza, circa 1 secondo (o 1 FPS su Raspi-Zero). Naturalmente, possiamo ridurre questa latenza e convertire il modello in TFLite o NCNN.

Rilevamento di Oggetti su un Live Streaming

Tutti i modelli esplorati in questo laboratorio possono rilevare oggetti in tempo reale utilizzando una telecamera. L'immagine catturata dovrebbe essere l'input per il modello addestrato e convertito. Per Raspi-4 o 5 con un desktop, OpenCV può catturare i frame e visualizzare il risultato dell'inferenza.

Tuttavia, è anche possibile creare un live streaming con una webcam per rilevare oggetti in tempo reale. Ad esempio, iniziamo con lo script sviluppato per l'app Image Classification e adattiamolo per un'*Applicazione Web di Rilevamento di Oggetti in Tempo Reale Utilizzando TensorFlow Lite e Flask*.

Questa versione dell'app funzionerà per tutti i modelli TFLite. Verificare che il modello sia nella cartella corretta, ad esempio:

```
model_path = "./models/ssd-mobilenet-v1-tflite-default-v1.tflite"
```

Scaricare lo script Python `object_detection_app.py` da GitHub.

E sul terminale, si esegue:

```
python3 object_detection_app.py
```

E accedere all'interfaccia web:

- Sul Raspberry Pi stesso (se si ha una GUI): si apre un browser web e si va su`http://localhost:5000`

- Da un altro dispositivo sulla stessa rete: aprire un browser web e andare su `http://<raspberry_pi_ip>:5000` (Sostituire `<raspberry_pi_ip>` con l'indirizzo IP del Raspberry Pi). Per esempio: `http://192.168.4.210:5000/`

Ecco alcuni screenshot dell'app in esecuzione su un desktop esterno

Diamo un'occhiata a una descrizione tecnica dei moduli chiave utilizzati nell'applicazione di rilevamento degli oggetti:

1. **TensorFlow Lite (tflite_runtime)**:
    - Scopo: Inferenza efficiente di modelli di machine learning su dispositivi edge.
    - Perché: TFLite offre dimensioni ridotte del modello e prestazioni ottimizzate rispetto a TensorFlow completo, il che è fondamentale per dispositivi con risorse limitate come Raspberry Pi. Supporta l'accelerazione hardware e la quantizzazione, migliorando ulteriormente l'efficienza.

- Funzioni chiave: `Interpreter` per caricare ed eseguire il modello, `get_input_details()` e `get_output_details()` per l'interfaccia col modello.

2. **Flask:**
   - Scopo: Framework web leggero per la creazione del server backend.
   - Perché: La semplicità e la flessibilità di Flask lo rendono ideale per lo sviluppo e la distribuzione rapidi di applicazioni web. Richiede meno risorse rispetto a framework più grandi, adatto a dispositivi edge.
   - Componenti chiave: decoratori di route per definire endpoint API, oggetti `Response` per lo streaming video, `render_template_string` per servire HTML dinamico.
3. **Picamera2:**
   - Scopo: Interfaccia con il modulo fotocamera Raspberry Pi.
   - Perché: Picamera2 è la libreria più recente per il controllo delle fotocamere Raspberry Pi, che offre prestazioni e funzionalità migliorate rispetto alla libreria Picamera originale.
   - Funzioni chiave: `create_preview_configuration()` per impostare la fotocamera, `capture_file()` per catturare i fotogrammi.
4. **PIL (Python Imaging Library):**
   - Scopo: Elaborazione e manipolazione delle immagini.
   - Perché: PIL fornisce un'ampia gamma di funzionalità di elaborazione delle immagini. Viene utilizzato qui per ridimensionare le immagini, disegnare riquadri di delimitazione e convertire tra formati di immagine.
   - Classi chiave: `Image` per caricare e manipolare immagini, `ImageDraw` per disegnare forme e testo sulle immagini.
5. **NumPy:**
   - Scopo: Operazioni array efficienti e calcolo numerico.
   - Perché: le operazioni su array di NumPy sono molto più veloci delle liste Python pure, il che è fondamentale per elaborare in modo efficiente i dati delle immagini e gli input/output del modello.
   - Funzioni chiave: `array()` per creare array, `expand_dims()` per aggiungere dimensioni agli array.
6. **Threading:**
   - Scopo: Esecuzione simultanea di attività.
   - Perché: Il threading consente l'acquisizione simultanea di frame, il rilevamento di oggetti e il funzionamento del server Web, cruciali per il mantenimento delle prestazioni in tempo reale.
   - Componenti chiave: La classe `Thread` crea thread di esecuzione separati e Lock viene utilizzato per la sincronizzazione dei thread.
7. **io.BytesIO:**
   - Scopo: Stream binari in memoria.
   - Perché: Consente una gestione efficiente dei dati delle immagini in memoria senza bisogno di file temporanei, migliorando la velocità e riducendo le operazioni di I/O.
8. **time:**
   - Scopo: Funzioni correlate al tempo.

    - Perché: Utilizzato per aggiungere ritardi (`time.sleep()`) per controllare la frequenza dei fotogrammi e per le misure delle prestazioni.
9. **jQuery (client-side)**:
    - Scopo: Manipolazione DOM semplificata e richieste AJAX.
    - Perché: Semplifica l'aggiornamento dinamico dell'interfaccia web e la comunicazione con il server senza ricaricamenti di pagina.
    - Funzioni chiave: `.get()` e `.post()` per richieste AJAX, metodi di manipolazione DOM per l'aggiornamento dell'interfaccia utente.

Per quanto riguarda l'architettura del sistema dell'app principale:

1. **Main Thread**: Esegue il server Flask, gestisce le richieste HTTP e serve l'interfaccia web.
2. **Camera Thread**: Cattura continuamente i frame dalla fotocamera.
3. **Detection Thread**: Elabora i frame tramite il modello TFLite per il rilevamento degli oggetti.
4. **Frame Buffer**: Spazio di memoria condiviso (protetto da blocchi) che memorizza i frame più recenti e i risultati del rilevamento.

E il flusso di dati dell'app, possiamo descriverlo in breve:

1. La fotocamera cattura il frame → Frame Buffer
2. Il thread di rilevamento legge dal Frame Buffer → Elabora tramite il modello TFLite → Aggiorna i risultati del rilevamento nel Frame Buffer
3. Flask indirizza l'accesso al Frame Buffer per fornire i risultati più recenti del frame e del rilevamento
4. Il client Web riceve gli aggiornamenti tramite AJAX e aggiorna l'interfaccia utente

Questa architettura consente un rilevamento efficiente degli oggetti in tempo reale, mantenendo al contempo un'interfaccia Web reattiva in esecuzione su un dispositivo edge con risorse limitate come un Raspberry Pi. Il threading e le librerie efficienti come TFLite e PIL consentono al sistema di elaborare i frame video in tempo reale, mentre Flask e jQuery forniscono un modo intuitivo per interagire con essi.
Si può testare l'app con un altro modello pre-elaborato, come EfficientDet, modificando la riga dell'app:

```
model_path = "./models/lite-model_efficientdet_lite0_detection_metadata_1.tflite"
```

> Per usare l'app per il modello SSD-MobileNetV2, addestrato su Edge Impulse Studio con il set di dati "Box versus Wheel", il codice dovrebbe essere adattato anche in base ai dettagli di input, come abbiamo esplorato sul suo notebook.

Conclusione

Questo laboratorio ha esplorato l'implementazione del rilevamento di oggetti su dispositivi edge come Raspberry Pi, dimostrando la potenza e il potenziale dell'esecuzione di attività avanzate di computer vision su hardware con risorse limitate. Abbiamo trattato diversi aspetti essenziali:

1. **Confronto tra Modelli**: Abbiamo esaminato diversi modelli di rilevamento oggetti, tra cui SSD-MobileNet, EfficientDet, FOMO e YOLO, confrontandone le prestazioni e i compromessi sui dispositivi edge.
2. **Training e Deployment**: Utilizzando un set di dati personalizzato di scatole e ruote (etichettato su Roboflow), abbiamo esaminato il processo di addestramento dei modelli utilizzando Edge Impulse Studio e Ultralytics e la loro distribuzione su Raspberry Pi.

3. **Tecniche di Ottimizzazione**: Per migliorare la velocità di inferenza sui dispositivi edge, abbiamo esplorato vari metodi di ottimizzazione, come la quantizzazione del modello (TFLite int8) e la conversione del formato (ad esempio, in NCNN).
4. **Applicazioni in Tempo Reale**: Il laboratorio ha esemplificato un'applicazione Web di rilevamento oggetti in tempo reale, dimostrando come questi modelli possono essere integrati in sistemi pratici e interattivi.
5. **Considerazioni sulle Prestazioni**: Durante il laboratorio abbiamo discusso l'equilibrio tra accuratezza del modello e velocità di inferenza, un aspetto fondamentale per le applicazioni di IA edge.

La capacità di eseguire il rilevamento di oggetti su dispositivi edge apre numerose possibilità in vari ambiti, dall'agricoltura di precisione, all'automazione industriale e al controllo di qualità, alle applicazioni per la casa intelligente e al monitoraggio ambientale. Elaborando i dati localmente, questi sistemi possono offrire latenza ridotta, privacy migliorata e funzionamento in ambienti con connettività limitata.

Guardando al futuro, le potenziali aree di ulteriore esplorazione includono: - Implementazione di pipeline multi-modello per attività più complesse - Esplorazione di opzioni di accelerazione hardware per Raspberry Pi - Integrazione del rilevamento di oggetti con altri sensori per sistemi AI edge più completi - Sviluppo di soluzioni edge-to-cloud che sfruttano sia l'elaborazione locale che le risorse cloud

Il rilevamento di oggetti su dispositivi edge può creare sistemi intelligenti e reattivi che portano la potenza dell'IA direttamente nel mondo fisico, aprendo nuove frontiere nel modo in cui interagiamo e comprendiamo il nostro ambiente.

Risorse

- Dataset ("Box versus Wheel")
- SSD-MobileNet Notebook on a Raspi
- EfficientDet Notebook on a Raspi
- FOMO - EI Linux Notebook on a Raspi
- YOLOv8 Box versus Wheel Dataset Training on CoLab
- Edge Impulse Project - SSD MobileNet and FOMO
- Script Python
- Modelli

Small Language Models (SLM)

DALL·E prompt - Un'illustrazione in stile cartoon anni '50 che mostra un Raspberry Pi che esegue un piccolo modello di linguaggio all'edge. Il Raspberry Pi è stilizzato in modo retro-futuristico con bordi arrotondati e cromature, collegato a sensori e dispositivi da cartone animato giocosi. I fumetti fluttuano in giro, rappresentando l'elaborazione del linguaggio, e lo sfondo ha un paesaggio stravagante di dispositivi interconnessi con fili e piccoli gadget, tutti disegnati in uno stile cartoon vintage. La tavolozza dei colori utilizza tenui colori pastello e contorni audaci tipici dei cartoni animati degli anni '50, conferendo un'atmosfera divertente e nostalgica alla scena.

Introduzione

Nell'area in rapida crescita dell'intelligenza artificiale, l'edge computing offre l'opportunità di decentralizzare le capacità tradizionalmente riservate a server potenti e centralizzati. Questo laboratorio esplora l'integrazione pratica di piccole versioni di modelli linguistici tradizionali di grandi dimensioni (LLM) in un Raspberry Pi 5, trasformando questo dispositivo edge in un hub di IA in grado di elaborare dati in tempo reale e in loco.

Man mano che i modelli linguistici di grandi dimensioni crescono in dimensioni e complessità, gli Small Language Model (SLM) offrono un'alternativa interessante per i dispositivi edge, raggiungendo un equilibrio tra prestazioni ed efficienza delle risorse. Eseguendo questi modelli direttamente su Raspberry Pi, possiamo creare applicazioni

reattive e rispettose della privacy che funzionano anche in ambienti con connettività Internet limitata o assente.
Questo laboratorio guiderà attraverso la configurazione, l'ottimizzazione e lo sfruttamento degli SLM su Raspberry Pi. Esploreremo l'installazione e l'utilizzo di Ollama. Questo framework open source ci consente di eseguire LLM localmente sulle nostre macchine (i nostri desktop o dispositivi edge come Raspberry Pi o NVidia Jetson). Ollama è progettato per essere efficiente, scalabile e facile da usare, il che lo rende una buona opzione per distribuire modelli di IA come Microsoft Phi, Google Gemma, Meta Llama e LLaVa (Multimodal). Integreremo alcuni di questi modelli in progetti utilizzando l'ecosistema Python, esplorandone il potenziale in scenari del mondo reale (o almeno indicheremo questa direzione).

Setup

Nei laboratori precedenti avremmo potuto usare qualsiasi modello Raspi, ma qui la scelta deve ricadere sul Raspberry Pi 5 (Raspi-5). È una piattaforma robusta che aggiorna sostanzialmente l'ultima versione 4, dotata di Broadcom BCM2712, una CPU Arm Cortex-A76 quad-core a 64 bit da 2,4 GHz con Cryptographic Extension e capacità di caching migliorate. Vanta una GPU VideoCore VII, due uscite HDMI® 4Kp60 con HDR e un decoder HEVC 4Kp60. Le opzioni di memoria includono 4 GB e 8 GB di SDRAM LPDDR4X ad alta velocità, con 8 GB come nostra scelta per eseguire SLM. Presenta inoltre un'archiviazione espandibile tramite uno slot per schede microSD e un'interfaccia PCIe 2.0 per periferiche veloci come gli SSD M.2 (Solid State Drive).

Per applicazioni SSL reali, gli SSD sono un'opzione migliore delle schede SD.

A proposito, come ha spiegato Alasdair Allan, l'inferenza diretta sulla CPU Raspberry Pi 5, senza accelerazione GPU, è ora alla pari con le prestazioni del Coral TPU.

Per maggiori informazioni, consultare l'articolo completo: Benchmarking TensorFlow e TensorFlow Lite su Raspberry Pi 5.

Raspberry Pi Active Cooler

Suggeriamo di installare un Active Cooler, una soluzione di raffreddamento clip-on dedicata per Raspberry Pi 5 (Raspi-5), per questo laboratorio. Combina un dissipatore di calore in alluminio con una ventola di raffreddamento a temperatura controllata per mantenere il Raspi-5 in funzione comodamente sotto carichi pesanti, come l'esecuzione di SLM.

L'Active Cooler ha dei pad termici pre-applicati per il trasferimento del calore ed è montato direttamente sulla scheda Raspberry Pi 5 tramite perni a molla. Il firmware Raspberry Pi lo gestisce attivamente: a 60 °C, la ventola del soffiatore viene accesa; a 67,5 °C, la velocità della ventola verrà aumentata; e infine, a 75 °C, la ventola aumenterà fino alla massima velocità. La ventola del soffiatore rallenterà automaticamente quando la temperatura scenderà al di sotto di questi limiti.

Per evitare il surriscaldamento, tutte le schede Raspberry Pi iniziano a limitare il processore quando la temperatura raggiunge gli 80 °C e a limitarlo ulteriormente quando raggiunge la temperatura massima di 85 °C (maggiori dettagli qui).

Generative AI (GenAI)

Generative AI è un sistema di IA in grado di creare nuovi contenuti originali su vari supporti come **testo, immagini, audio e video**. Questi sistemi apprendono modelli da dati esistenti e utilizzano tale conoscenza per generare nuovi output che in precedenza non esistevano. **Large Language Models (LLM)**, **Small Language Models (SLM)** e **modelli multimodali** possono essere tutti considerati tipi di GenAI quando utilizzati per attività generative.
GenAI fornisce il framework concettuale per la creazione di contenuti basati sull'intelligenza artificiale, con gli LLM che fungono da potenti generatori di testo per uso generale. Gli SLM adattano questa tecnologia all'edge computing, mentre i modelli multimodali estendono le capacità di GenAI a diversi tipi di dati. Insieme, rappresentano uno spettro di tecnologie di intelligenza artificiale generativa, ciascuna con i suoi punti di forza e applicazioni, che guidano collettivamente la creazione e la comprensione di contenuti basati sull'IA.

Large Language Models (LLM)

I "Large Language Model (LLM)" sono sistemi avanzati di intelligenza artificiale che comprendono, elaborano e generano testi simili a quelli umani. Questi modelli sono caratterizzati dalla loro enorme scala in termini di quantità di dati su cui vengono addestrati e numero di parametri che contengono. Gli aspetti critici degli LLM includono:

1. **Dimensioni**: Gli LLM contengono in genere miliardi di parametri. Ad esempio, GPT-3 ha 175 miliardi di parametri, mentre alcuni modelli più recenti superano un trilione di parametri.
2. **Dati di Addestramento**: Vengono addestrati su grandi quantità di dati di testo, spesso inclusi libri, siti Web e altre fonti diverse, che ammontano a centinaia di gigabyte o persino terabyte di testo.
3. **Architettura**: La maggior parte degli LLM utilizza architetture basate su trasformatori, che consentono loro di elaborare e generare testo prestando attenzione a diverse parti dell'input contemporaneamente.
4. **Capacità**: Gli LLM possono eseguire un'ampia gamma di attività linguistiche senza una messa a punto specifica, tra cui:
    - Generazione di testo
    - Traduzione
    - Riepilogo
    - Risposte a domande
    - Generazione di codice
    - Ragionamento logico
5. **Apprendimento a Intervalli**: Spesso possono comprendere ed eseguire nuove attività con esempi o istruzioni minimi.
6. **Richiede molte Risorse**: A causa delle loro dimensioni, gli LLM richiedono in genere risorse di elaborazione significative per funzionare, spesso necessitando di potenti GPU o TPU.
7. **Sviluppo Continuo**: Il campo degli LLM è in rapida evoluzione, con nuovi modelli e tecniche che emergono costantemente.
8. **Considerazioni Etiche**: L'uso degli LLM solleva importanti questioni su pregiudizi, disinformazione e impatto ambientale della formazione di modelli così grandi.

9. **Applicazioni**: Gli LLM sono utilizzati in vari campi, tra cui la creazione di contenuti, il servizio clienti, l'assistenza alla ricerca e lo sviluppo di software.
10. **Limitazioni**: Nonostante la loro potenza, gli LLM possono produrre informazioni errate o distorte e non hanno capacità di vera comprensione o ragionamento.

Dobbiamo notare che utilizziamo modelli di grandi dimensioni oltre al testo, chiamandoli *modelli multimodali*. Questi modelli integrano ed elaborano informazioni da più tipi di input contemporaneamente. Sono progettati per comprendere e generare contenuti in varie forme di dati, come testo, immagini, audio e video.
Certamente. Definiamo modelli "open" [aperti] e "closed" [chiusi] nel contesto di modelli di linguaggio e IA:

Modelli Chiusi e Aperti:

Modelli Chiusi, detti anche modelli proprietari, sono modelli di IA il cui funzionamento interno, codice e dati di addestramento non sono divulgati pubblicamente. Esempi: GPT-4 (di OpenAI), Claude (di Anthropic), Gemini (di Google).

Modelli Aperti, noti anche come modelli open source, sono modelli di intelligenza artificiale il cui codice sottostante, architettura e spesso dati di addestramento sono disponibili e accessibili al pubblico. Esempi: Gemma (di Google), LLaMA (di Meta) e Phi (di Microsoft).
I modelli aperti sono particolarmente rilevanti per l'esecuzione di modelli su dispositivi edge come Raspberry Pi in quanto possono essere più facilmente adattati, ottimizzati e distribuiti in ambienti con risorse limitate. Tuttavia, è fondamentale verificare le loro Licenze. I modelli aperti sono dotati di varie licenze open source che possono influenzare il loro utilizzo in applicazioni commerciali, mentre i modelli chiusi hanno termini di servizio chiari, seppur restrittivi.

Adattato da https://arxiv.org/pdf/2304.13712

Small Language Models (SLM)

Nel contesto dell'edge computing su dispositivi come Raspberry Pi, gli LLM su larga scala sono in genere troppo grandi e dispendiosi in termini di risorse per essere eseguiti direttamente. Questa limitazione ha spinto lo sviluppo di modelli più piccoli ed efficienti, come gli Small Language Models (SLM).
Gli SLM sono versioni compatte degli LLM progettate per essere eseguite in modo efficiente su dispositivi con risorse limitate come smartphone, dispositivi IoT e computer a scheda singola come Raspberry Pi. Questi modelli sono significativamente più piccoli in termini di

dimensioni e requisiti di calcolo rispetto alle loro controparti più grandi, pur mantenendo impressionanti capacità di comprensione e generazione del linguaggio.
Le caratteristiche principali degli SLM includono:

1. **Conteggio ridotto dei parametri**: In genere vanno da poche centinaia di milioni a qualche miliardo di parametri, rispetto ai miliardi a due cifre nei modelli più grandi.
2. **Meno ingombro di memoria**: Richiedono, al massimo, pochi gigabyte di memoria anziché decine o centinaia di gigabyte.
3. **Tempo di inferenza più rapido**: Possono generare risposte in millisecondi o secondi su dispositivi edge.
4. **Efficienza energetica**: Consumano meno energia, rendendoli adatti per dispositivi alimentati a batteria.
5. **Tutela della privacy**: Abilitano l'elaborazione sul dispositivo senza inviare dati ai server cloud.
6. **Funzionalità offline**: Funzionano senza una connessione Internet.

Gli SLM raggiungono le loro dimensioni compatte attraverso varie tecniche come la distillazione della conoscenza, la potatura del modello e la quantizzazione. Sebbene non possano eguagliare le ampie capacità dei modelli più grandi, gli SLM eccellono in compiti e domini specifici, il che li rende ideali per applicazioni mirate su dispositivi edge.

> In genere prenderemo in considerazione gli SLM, modelli linguistici con meno di 5 miliardi di parametri quantizzati a 4 bit.

Esempi di SLM includono versioni compresse di modelli come Meta Llama, Microsoft PHI e Google Gemma. Questi modelli consentono un'ampia gamma di attività di elaborazione del linguaggio naturale direttamente sui dispositivi edge, dalla classificazione del testo e analisi del sentiment alle risposte alle domande e alla generazione di testo limitato.
Per maggiori informazioni sugli SLM, il documento, LLM Pruning and Distillation in Practice: The Minitron Approach, fornisce un approccio che applica il pruning e la distillazione per ottenere SLM da LLM. E, SMALL LANGUAGE MODELS: SURVEY, MEASUREMENTS, AND INSIGHTS, presenta un'indagine e un'analisi complete di Small Language Models (SLM), che sono modelli linguistici con da 100 milioni a 5 miliardi di parametri progettati per dispositivi con risorse limitate.

Ollama

logo di ollama

Ollama è un framework open source che ci consente di eseguire modelli linguistici (LM), grandi o piccoli, localmente sulle nostre macchine. Ecco alcuni punti critici su Ollama:

1. **Esecuzione del Modello Locale**: Ollama consente di eseguire LM su personal computer o dispositivi edge come Raspi-5, eliminando la necessità di chiamate API basate su cloud.

2. **Facilità d'Uso**: Fornisce una semplice interfaccia a riga di comando per scaricare, eseguire e gestire diversi modelli linguistici.
3. **Varietà di Modelli**: Ollama supporta vari LLM, tra cui Phi, Gemma, Llama, Mistral e altri modelli open source.
4. **Personalizzazione**: Gli utenti possono creare e condividere modelli personalizzati su misura per esigenze o domini specifici.
5. **Leggero**: Progettato per essere efficiente e funzionare su hardware di livello consumer.
6. **Integrazione API**: Offre un'API che consente l'integrazione con altre applicazioni e servizi.
7. **Incentrato sulla Privacy**: Eseguendo i modelli localmente, affronta i problemi di privacy associati all'invio di dati a server esterni.
8. **Multipiattaforma**: Disponibile per sistemi macOS, Windows e Linux (il nostro caso, qui).
9. **Sviluppo Attivo**: Aggiornato regolarmente con nuove funzionalità e supporto per i modelli.
10. **Guidato dalla Community**: Trae vantaggio dai contributi della community e dalla condivisione dei modelli.

Per saperne di più su cosa sia Ollama e come funziona sotto internamente, si può guardare questo breve video di Matt Williams, uno dei fondatori di Ollama:
https://www.youtube.com/embed/90ozfdsQOKo

> Matt ha un corso completamente gratuito su Ollama che consigliamo:
> https://youtu.be/9KEUFe4KQAI?si=D_-q3CMbHiT-twuy

Installazione di Ollama

Impostiamo e attiviamo un ambiente virtuale per lavorare con Ollama:

```
python3 -m venv ~/ollama
source ~/ollama/bin/activate
```

Ed eseguiamo il comando per installare Ollama:

```
curl -fsSL https://ollama.com/install.sh | sh
```

Di conseguenza, un'API verrà eseguita in background su `127.0.0.1:11434`. D'ora in poi, possiamo eseguire Ollama tramite il terminale. Per iniziare, verifichiamo la versione di Ollama, che ci dirà anche se è installata correttamente:

```
ollama -v
```

Nella pagina della Libreria Ollama, possiamo trovare i modelli supportati da Ollama. Ad esempio, filtrando per `Most popular`, possiamo vedere Meta Llama, Google Gemma, Microsoft Phi, LLaVa, ecc.

Meta Llama 3.2 1B/3B

Installiamo ed eseguiamo il nostro primo piccolo modello linguistico, Llama 3.2 1B (e 3B). La serie Meta Llama 3.2 comprende un set di modelli linguistici generativi multilingue disponibili in dimensioni di parametri pari a 1 miliardo e 3 miliardi. Questi modelli sono progettati per elaborare input di testo e generare output di testo. Le varianti sintonizzate sulle istruzioni all'interno di questa raccolta sono specificamente ottimizzate per applicazioni conversazionali multilingue, tra cui attività che comportano il recupero e la sintesi delle informazioni con un approccio agentico. Rispetto a molti modelli di chat open source e proprietari esistenti, i modelli sintonizzati sulle istruzioni Llama 3.2 dimostrano prestazioni superiori su benchmark di settore ampiamente utilizzati.
I modelli 1B e 3B sono stati potati da Llama 8B, quindi i logit dai modelli 8B e 70B sono stati utilizzati come target a livello di token (distillazione a livello di token). La distillazione della conoscenza è stata utilizzata per recuperare le prestazioni (sono stati addestrati con 9 trilioni di token). Il modello 1B ha 1,24B, quantizzati a intero (Q8_0), e i parametri 3B, 3,12B, con una quantizzazione Q4_0, che termina con una dimensione di 1,3 GB e 2 GB, rispettivamente. La sua finestra di contesto è di 131.072 token.

Installare ed eseguire il Model

```
ollama run llama3.2:1b
```

Eseguendo il modello col comando precedente, dovremmo avere il prompt di Ollama disponibile per inserire una domanda e iniziare a chattare con il modello LLM; ad esempio,

```
>>> What is the capital of France?
```

Quasi immediatamente, otteniamo la risposta corretta:

```
The capital of France is Paris.
```

Utilizzando l'opzione `--verbose` quando si richiama il modello, verranno generate diverse statistiche sulle sue prestazioni (Il modello eseguirà il polling solo la prima volta che eseguiamo il comando).

```
marcelo_rovai — mjrovai@raspi-5: ~ — ssh mjrovai@192.168.4.209 — 79×26
(ollama) mjrovai@raspi-5:~ $ ollama run llama3.2:1b --verbose
pulling manifest
pulling 74701a8c35f6... 100%  1.3 GB
pulling 966de95ca8a6... 100%  1.4 KB
pulling fcc5a6bec9da... 100%  7.7 KB
pulling a70ff7e570d9... 100%  6.0 KB
pulling 4f659a1e86d7... 100%  485 B
verifying sha256 digest
writing manifest
success
>>> What is the capital of France?
The capital of France is Paris.

total duration:       2.620170326s
load duration:        39.947908ms
prompt eval count:    32 token(s)
prompt eval duration: 1.644773s
prompt eval rate:     19.46 tokens/s
eval count:           8 token(s)
eval duration:        889.941ms
eval rate:            8.99 tokens/s
```

Ogni metrica fornisce informazioni su come il modello elabora gli input e genera gli output. Ecco una ripartizione del significato di ogni metrica:

- **Total Duration (2.620170326s)**: Questo è il tempo completo impiegato dall'inizio del comando al completamento della risposta. Comprende il caricamento del modello, l'elaborazione del prompt di input e la generazione della risposta.
- **Load Duration (39.947908ms)**: Questa durata indica il tempo necessario per caricare il modello o i componenti necessari nella memoria. Se questo valore è minimo, può suggerire che il modello è stato precaricato o che è stata richiesta solo una configurazione minima.
- **Prompt Eval Count (32 tokens)**: Il numero di token nel prompt di input. In NLP, i token sono in genere parole o sotto-parole, quindi questo conteggio include tutti i token che il modello ha valutato per comprendere e rispondere alla query.
- **Prompt Eval Duration (1.644773s)**: Misura il tempo impiegato dal modello per valutare o elaborare il prompt di input. Rappresenta la maggior parte della durata totale, il che implica che comprendere la query e preparare una risposta è la parte più dispendiosa in termini di tempo del processo.
- **Prompt Eval Rate (19.46 tokens/s)**: Questa frequenza indica la rapidità con cui il modello elabora i token dal prompt di input. Riflette la velocità del modello in termini di comprensione del linguaggio naturale.

- **Eval Count (8 token(s))**: Questo è il numero di token nella risposta del modello, che in questo caso era "The capital of France is Paris."
- **Eval Duration (889.941ms)**: Questo è il tempo impiegato per generare l'output in base all'input valutato. È molto più breve della valutazione del prompt, il che suggerisce che generare la risposta è meno complesso o computazionalmente intensivo rispetto alla comprensione del prompt.
- **Eval Rate (8.99 tokens/s)**: Simile alla frequenza di valutazione del prompt, indica la velocità con cui il modello genera token di output. È una metrica fondamentale per comprendere l'efficienza del modello nella generazione di output.

Questa ripartizione dettagliata può aiutare a comprendere le richieste computazionali e le caratteristiche delle prestazioni dell'esecuzione di SLM come Llama su dispositivi edge come Raspberry Pi 5. Mostra che mentre la valutazione del prompt richiede più tempo, la generazione effettiva delle risposte è relativamente più rapida. Questa analisi è fondamentale per ottimizzare le prestazioni e diagnosticare potenziali colli di bottiglia nelle applicazioni in tempo reale.
Caricando ed eseguendo il modello 3B, possiamo vedere la differenza nelle prestazioni per lo stesso prompt;

```
marcelo_rovai — mjrovai@raspi-5: ~ — ssh mjrovai@192.168.4.209 — 74×12
(ollama) mjrovai@raspi-5:~ $ ollama run llama3.2:3b --verbose
>>> What is the capital of France?
The capital of France is Paris.

total duration:       1.808927736s
load duration:        39.854862ms
prompt eval count:    32 token(s)
prompt eval duration: 221.506ms
prompt eval rate:     144.47 tokens/s
eval count:           8 token(s)
eval duration:        1.506376s
eval rate:            5.31 tokens/s
```

Il tasso di valutazione è inferiore, 5,3 token/s rispetto ai 9 token/s del modello più piccolo.
Quando si chiede

```
>>> What is the distance between Paris and Santiago, Chile?
```

Il modello 1B ha risposto `9,841 kilometers (6,093 miles)`, il che è impreciso, e il modello 3B ha risposto `7,300 miles (11,700 km)`, il che è vicino alla distanza corretta (11,642 km).
Chiediamo le coordinate di Parigi:

```
>>> what is the latitude and longitude of Paris?
The latitude and longitude of Paris are 48.8567° N (48°55'
42" N) and 2.3510° E (2°22' 8" E), respectively.
```

Sia i modelli 1B che 3B hanno dato risposte corrette.

Google Gemma 2 2B

Installiamo Gemma 2, un modello efficiente e ad alte prestazioni disponibile in tre dimensioni: 2B, 9B e 27B. Installeremo **Gemma 2 2B**, un modello leggero addestrato con 2 trilioni di token che produce risultati sproporzionati imparando da modelli più grandi tramite distillazione. Il modello ha 2,6 miliardi di parametri e una quantizzazione Q4_0, che termina con una dimensione di 1,6 GB. La sua finestra di contesto è di 8.192 token.

Installare ed eseguire il Model

```
ollama run gemma2:2b --verbose
```

Eseguendo il modello col comando precedente, dovremmo avere il prompt di Ollama disponibile per inserire una domanda e iniziare a chattare con il modello LLM; ad esempio,

```
>>> What is the capital of France?
```

Quasi immediatamente, otteniamo la risposta corretta:

```
The capital of France is **Paris**. 🗼
```

E le statistiche.

Possiamo vedere che Gemma 2:2B ha più o meno le stesse prestazioni di Lama 3.2:3B, ma con meno parametri.

Altri esempi:

```
>>> What is the distance between Paris and Santiago, Chile?

The distance between Paris, France and Santiago, Chile is
approximately **7,000 miles (11,267 kilometers)**.

Keep in mind that this is a straight-line distance, and actual
travel distance can vary depending on the chosen routes and any
stops along the way. 🛫`
```

Inoltre, una buona risposta ma meno precisa di Llama3.2:3B.

```
>>> what is the latitude and longitude of Paris?

You got it! Here are the latitudes and longitudes of Paris,
France:

* **Latitude:** 48.8566° N (north)
* **Longitude:** 2.3522° E (east)
```

```
Let me know if you'd like to explore more about Paris or its
location! 🗼FR
```

Una risposta buona e precisa (un po' più prolissa delle risposte del lama).

Microsoft Phi3.5 3.8B

Prendiamo un modello più grande (ma comunque "tiny"), il PHI3.5, un modello aperto all'avanguardia leggero da 3,8B di Microsoft. Il modello appartiene alla famiglia di modelli Phi-3 e supporta la lunghezza del contesto `token 128K` e le lingue: arabo, cinese, ceco, danese, olandese, inglese, finlandese, francese, tedesco, ebraico, ungherese, italiano, giapponese, coreano, norvegese, polacco, portoghese, russo, spagnolo, svedese, tailandese, turco e ucraino.

La dimensione del modello, in termini di byte, dipenderà dal formato di quantizzazione specifico utilizzato. La dimensione può variare dalla quantizzazione a 2 bit (`q2_k`) di 1,4 GB (prestazioni più elevate/qualità inferiore) alla quantizzazione a 16 bit (fp-16) di 7,6 GB (prestazioni più basse/qualità più elevata).

Eseguiamo la quantizzazione a 4 bit (`Q4_0`), che richiederà 2,2 GB di RAM, con un compromesso intermedio per quanto riguarda la qualità dell'output e le prestazioni.

```
ollama run phi3.5:3.8b --verbose
```

> Si può usare `run` o `pull` per scaricare il modello. Ciò che accade è che Ollama tiene nota dei modelli estratti e, una volta che PHI3 non esiste, prima di eseguirlo, Ollama lo estrae.

Immettiamo lo stesso prompt usato prima:

```
>>> What is the capital of France?


The capital of France is Paris. It' extradites significant
historical, cultural, and political importance to the country as
well as being a major European city known for its art, fashion,
gastronomy, and culture. Its influence extends beyond national
borders, with millions of tourists visiting each year from around
the globe. The Seine River flows through Paris before it reaches
the broader English Channel at Le Havre. Moreover, France is one
of Europe's leading economies with its capital playing a key role

...
```

La risposta è stata molto "verbosa", specifichiamo un prompt migliore:

In questo caso, la risposta è stata comunque più lunga di quanto ci aspettassimo, con una velocità di valutazione di 2,25 token/s, più del doppio di quella di Gemma e Llama.

> Scegliere il prompt più appropriato è una delle competenze più importanti da utilizzare con gli LLM, indipendentemente dalle dimensioni.

Quando abbiamo posto le stesse domande su distanza e latitudine/longitudine, non abbiamo ottenuto una buona risposta per una distanza di `13,507 kilometers (8,429 miles)`, ma andava bene per le coordinate. Di nuovo, avrebbe potuto essere meno verbosa (più di 200 token per ogni risposta).
Possiamo usare qualsiasi modello come assistente poiché la loro velocità è relativamente decente, ma al 24 settembre (2023), Llama2:3B è una scelta migliore. Si dovrebbero provare altri modelli, a seconda delle esigenze. 🤗 Open LLM Leaderboard può dare un'idea sui migliori modelli in termini di dimensioni, benchmark, licenza, ecc.

> Il miglior modello da usare è quello adatto alle specifiche necessità. Inoltre, si tenga presente che questo campo si evolve con nuovi modelli ogni giorno,

Modelli Multimodali

I modelli multimodali sono sistemi di intelligenza artificiale (IA) in grado di elaborare e comprendere informazioni provenienti da più fonti, come immagini, testo, audio e video. Nel nostro contesto, gli LLM multimodali possono elaborare vari input, tra cui testo, immagini e audio, come prompt e convertire tali prompt in vari output, non solo il tipo di sorgente.
Qui lavoreremo con LLaVA-Phi-3, un modello LLaVA ottimizzato da Phi 3 Mini 4k. Ha solidi benchmark di prestazioni che sono alla pari con il modello originale LLaVA (Large Language and Vision Assistant).
LLaVA-Phi-3 è un modello multimodale di grandi dimensioni addestrato end-to-end progettato per comprendere e generare contenuti in base a input visivi (immagini) e istruzioni testuali. Combina le capacità di un codificatore visivo e di un modello linguistico per elaborare e rispondere a input multimodali.
Installiamo il modello:

```
ollama run llava-phi3:3.8b --verbose
```

Cominciamo con un input di testo:

```
>>> You are a helpful AI assistant. What is the capital of France?

As an AI language model, I can tell you that the capital of France
is Paris. It's not only the largest city in the country but also
serves as its political and administrative center. Paris is known
for its iconic landmarks such as the Eiffel Tower, Notre-Dame
Cathedral, and the Louvre Museum. The city has a rich history,
beautiful architecture, and is widely considered to be one of the
most romantic cities in the world.
```

La risposta ha richiesto circa 30 secondi, con una velocità di valutazione di 3,93 token/s! Niente male!
Ma vediamo di immettere un'immagine come input. Per questo, creiamo una directory per lavorare:

```
cd Documents/
mkdir OLLAMA
cd OLLAMA
```

Scarichiamo un'immagine 640x320 da Internet, per esempio (Wikipedia: Paris, France):

Utilizzando FileZilla, ad esempio, carichiamo l'immagine nella cartella OLLAMA sul Raspi-5 e chiamiamola `image_test_1.jpg`. Dovremmo avere l'intero path dell'immagine (possiamo usare `pwd` per ottenerlo).

`/home/mjrovai/Documents/OLLAMA/image_test_1.jpg`

Se si usa un desktop, si può copiare il path dell'immagine cliccando sull'immagine con il tasto destro del mouse.

Immettiamo questo prompt:

```
>>> Describe the image /home/mjrovai/Documents/OLLAMA/image_test_1.jpg
```

Il risultato è stato ottimo, ma la latenza complessiva è stata significativa: quasi 4 minuti per eseguire l'inferenza.

```
marcelo_rovai — mjrovai@raspi-5: ~/Documents/OLLAMA — ssh mjrovai@192.168.4.209 — 84×36
(ollama) mjrovai@raspi-5:~/Documents/OLLAMA $ pwd
/home/mjrovai/Documents/OLLAMA
(ollama) mjrovai@raspi-5:~/Documents/OLLAMA $ ollama run llava-phi3:3.8b --verbose
>>> Describe the image /home/mjrovai/Documents/OLLAMA/image_test_1.jpg
Added image '/home/mjrovai/Documents/OLLAMA/image_test_1.jpg'
The image captures a breathtaking view of Paris, France. The cityscape is
dotted with buildings in various shades of white and gray, interspersed with
lush green trees that add a touch of nature to the urban setting.

In the heart of the scene stands the Eiffel Tower, an iconic symbol of Paris,
its iron lattice structure reaching up into the clear blue sky. The tower's
distinctive silhouette is unmistakable against the backdrop of the sky, which
is a vibrant shade of blue with just a few clouds scattered across it.

The Seine River gracefully winds its way through the city, bordered by an
array of buildings on both sides. The river is lined with several bridges that
connect different parts of the city and facilitate movement for pedestrians
and vehicles alike.

Above all these elements, a few birds can be seen soaring freely in the sky,
their presence adding life to the scene. Their flight paths crisscross over
the river and the buildings, creating dynamic patterns that draw the eye.

Overall, this image presents a beautiful daytime snapshot of Paris - its
architectural marvels, natural beauty, and bustling city life coexisting in
harmony.

total duration:       3m55.972199346s
load duration:        16.198011ms
prompt eval count:    1 token(s)
prompt eval duration: 2m19.561783s
prompt eval rate:     0.01 tokens/s
eval count:           276 token(s)
eval duration:        1m36.330959s
eval rate:            2.87 tokens/s
>>> Send a message (/? for help)
```

Ispezione delle risorse locali

Utilizzando htop, possiamo monitorare le risorse in esecuzione sul nostro dispositivo.

```
htop
```

Durante il periodo in cui il modello è in esecuzione, possiamo ispezionare le risorse:

```
marcelo_rovai — mjrovai@raspi-5: ~ — ssh mjrovai@192.168.4.209 — 80×24
  0[||||||||||||||||||||||||||100.0%] Tasks: 73, 152 thr, 120 kthr; 4 runnin
  1[||||||||||||||||||||||||||98.7%] Load average: 0.56 0.36 0.17
  2[||||||||||||||||||||||||||98.0%] Uptime: 01:15:16
  3[||||||||||||||||||||||||||98.7%]
Mem[||||||||||||||||||||||3.23G/7.86G]
Swp[                          0K/200M]

 Main   I/O
  PID USER      PRI  NI  VIRT   RES   SHR S  CPU%-MEM%   TIME+  Command
 2951 ollama     20   0 3991M 2927M  5632 R 387.5 36.4  2:13.23 /tmp/ollama84
 2646 mjrovai    20   0 2744M 24064 15872 S   1.3  0.3  0:01.01 ollama run ge
 2665 mjrovai    20   0 2744M 24064 15872 S   1.3  0.3  0:00.20 ollama run ge
 2443 ollama     20   0 3753M  886M  873M S   0.7 11.0  0:45.54 /usr/local/bi
 2542 ollama     20   0 3753M  886M  873M S   0.7 11.0  0:01.68 /usr/local/bi
 3163 mjrovai    20   0  8112  3584  2560 R   0.7  0.0  0:00.17 htop
 3196 ollama     20   0 3991M 2927M  5632 R   0.7 36.4  0:00.01 /tmp/ollama84
    1 root       20   0  166M 13232  8192 S   0.0  0.2  0:01.59 /sbin/init sp
  299 root       20   0 66944 18528 16992 S   0.0  0.2  0:00.31 /lib/systemd/
  331 root       20   0 26864  6656  4096 S   0.0  0.1  0:00.18 /lib/systemd/
  643 systemd-ti 20   0 90944  6656  5632 S   0.0  0.1  0:00.06 /lib/systemd/
  663 systemd-ti 20   0 90944  6656  5632 S   0.0  0.1  0:00.00 /lib/systemd/
  664 root       20   0  232M  6656  5632 S   0.0  0.1  0:00.10 /usr/libexec/
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Tutte e quattro le CPU funzionano a quasi il 100% della loro capacità e la memoria utilizzata con il modello caricato è di `3.24GB`. Uscendo da Ollama, la memoria scende a circa `377 MB` (senza desktop).

È anche essenziale monitorare la temperatura. Quando si esegue Raspberry con un desktop, è possibile visualizzare la temperatura sulla barra delle applicazioni:

Se si è "headless", la temperatura può essere monitorata col comando:

```
vcgencmd measure_temp
```

Se non si fa nulla, la temperatura è di circa `50°C` per le CPU in esecuzione all'1%. Durante l'inferenza, con le CPU al 100%, la temperatura può salire fino a quasi `70°C`. Questo è OK e significa che il dissipatore attivo funziona, mantenendo la temperatura al di sotto di 80°C / 85°C (il suo limite).

Libreria Python Ollama

Finora, abbiamo esplorato la capacità di chat degli SLM utilizzando la riga di comando su un terminale. Tuttavia, vogliamo integrare quei modelli nei nostri progetti, quindi Python sembra essere la strada giusta. La buona notizia è che Ollama ha una libreria del genere.
La libreria Python Ollama semplifica l'interazione con modelli LLM avanzati, consentendo risposte e capacità più sofisticate, oltre a fornire il modo più semplice per integrare progetti Python 3.8+ con Ollama.
Per una migliore comprensione di come creare app usando Ollama con Python, possiamo seguire i video di Matt Williams, come quello qui sotto:
https://www.youtube.com/embed/_4K20tOsXK8

Installazione:
Nel terminale, eseguire il comando:

```
pip install ollama
```

Avremo bisogno di un editor di testo o di un IDE per creare uno script Python. Se si esegue Raspberry OS su un desktop, diverse opzioni, come Thonny e Geany, sono già state installate di default (accessibili tramite `[Menu][Programming]`). Si possono scaricare altri IDE, come Visual Studio Code, da `[Menu][Recommended Software]`. Quando si apre la finestra, si va su `[Programming]`, e si seleziona l'opzione che si preferisce e si preme `[Apply]`.

Se si preferisce usare Jupyter Notebook per lo sviluppo:

```
pip install jupyter
jupyter notebook --generate-config
```

Per eseguire Jupyter Notebook, si lancia il comando (cambiare l'indirizzo IP per il proprio):

```
jupyter notebook --ip=192.168.4.209 --no-browser
```

Sul terminale, si può vedere l'indirizzo URL locale per aprire il notebook:

Possiamo accedervi da un altro computer inserendo l'indirizzo IP del Raspberry Pi e il token fornito in un browser web (dovremmo copiarlo dal terminale).
Nella nostra directory di lavoro nel Raspi, creeremo un nuovo notebook Python 3.
Entriamo con uno script molto semplice per verificare i modelli installati:

```
import ollama
ollama.list()
```

Tutti i modelli verranno stampati come un dizionario, ad esempio:

```
{'name': 'gemma2:2b',
   'model': 'gemma2:2b',
   'modified_at': '2024-09-24T19:30:40.053898094+01:00',
   'size': 1629518495,
   'digest': '8ccf136fdd5298f3ffe2d69862750ea7fb56555fa4d5b18c04e3fa4d82ee09d7',
   'details': {'parent_model': '',
    'format': 'gguf',
    'family': 'gemma2',
    'families': ['gemma2'],
    'parameter_size': '2.6B',
    'quantization_level': 'Q4_0'}}]}
```

Ripetiamo una delle domande che abbiamo fatto prima, ma ora usando `ollama.generate()` dalla libreria Python Ollama. Questa API genererà una risposta per il prompt specificato con il modello fornito. Questo è un endpoint di streaming, quindi ci saranno una serie di risposte. L'oggetto di risposta finale includerà statistiche e dati aggiuntivi dalla richiesta.

```
MODEL = 'gemma2:2b'
PROMPT = 'What is the capital of France?'

res = ollama.generate(model=MODEL, prompt=PROMPT)
print (res)
```

Nel caso in cui si stia eseguendo il codice come script Python, lo si deve salvare, ad esempio, test_ollama.py. Si può usare l'IDE per eseguirlo o farlo direttamente sul terminale. Da ricordare, inoltre, che si deve sempre chiamare il modello e definirlo quando si esegue uno script stand-alone.

```
python test_ollama.py
```

Di conseguenza, avremo la risposta del modello in un formato JSON:

```
{'model': 'gemma2:2b', 'created_at': '2024-09-25T14:43:31.869633807Z',
'response': 'The capital of France is **Paris**. FR \n', 'done': True,
'done_reason': 'stop', 'context': [106, 1645, 108, 1841, 603, 573, 6037, 576,
6081, 235336, 107, 108, 106, 2516, 108, 651, 6037, 576, 6081, 603, 5231, 29437,
168428, 235248, 244304, 241035, 235248, 108], 'total_duration': 24259469458,
'load_duration': 19830013859, 'prompt_eval_count': 16, 'prompt_eval_duration':
1908757000, 'eval_count': 14, 'eval_duration': 2475410000}
```

Come possiamo vedere, vengono generate diverse informazioni, come:

- **response**: il testo di output principale generato dal modello in risposta al nostro prompt.
  - `The capital of France is **Paris**. FR`
- **context**: gli ID token che rappresentano l'input e il contesto utilizzati dal modello. I token sono rappresentazioni numeriche del testo utilizzate per l'elaborazione dal modello linguistico.
  - `[106, 1645, 108, 1841, 603, 573, 6037, 576, 6081, 235336, 107, 108, 106, 2516, 108, 651, 6037, 576, 6081, 603, 5231, 29437, 168428, 235248, 244304, 241035, 235248, 108]`

Le Metriche delle Prestazioni:

- **total_duration**: Tempo totale impiegato per l'operazione in nanosecondi. In questo caso, circa 24,26 secondi.
- **load_duration**: Tempo impiegato per caricare il modello o i componenti in nanosecondi. Circa 19,83 secondi.
- **prompt_eval_duration**: Tempo impiegato per valutare il prompt in nanosecondi. Circa 16 nanosecondi.
- **eval_count**: Numero di token valutati durante la generazione. Qui, 14 token.
- **eval_duration**: Tempo impiegato dal modello per generare la risposta in nanosecondi. Circa 2,5 secondi.

Ma ciò che vogliamo è la semplice 'response' e, forse per l'analisi, la durata totale dell'inferenza, quindi modifichiamo il codice per estrarlo dal dizionario:

```
print(f"\n{res['response']}")
print(f"\n [INFO] Total Duration: {(res['total_duration']/1e9):.2f} seconds")
```

Ora, abbiamo:

```
The capital of France is **Paris**. FR

 [INFO] Total Duration: 24.26 seconds
```

Utilizzo di Ollama.chat()

Un altro modo per ottenere la nostra risposta è utilizzare `ollama.chat()`, che genera il messaggio successivo in una chat con un modello fornito. Questo è un endpoint di streaming, quindi si verificheranno una serie di risposte. Lo streaming può essere disabilitato utilizzando `"stream": false`. L'oggetto di risposta finale includerà anche statistiche e dati aggiuntivi dalla richiesta.

```
PROMPT_1 = 'What is the capital of France?'

response = ollama.chat(model=MODEL, messages=[
{'role': 'user','content': PROMPT_1,},])
resp_1 = response['message']['content']
print(f"\n{resp_1}")
print(f"\n [INFO] Total Duration: {(res['total_duration']/1e9):.2f} seconds")
```

La risposta è la stessa di prima.

Una considerazione importante è che usando `ollama.generate()`, la risposta è "chiara" dalla "memoria" del modello dopo la fine dell'inferenza (usata solo una volta), ma se vogliamo mantenere una conversazione, dobbiamo usare `ollama.chat()`. Vediamolo in azione:

```
PROMPT_1 = 'What is the capital of France?'
response = ollama.chat(model=MODEL, messages=[
{'role': 'user','content': PROMPT_1,},])
resp_1 = response['message']['content']
print(f"\n{resp_1}")
print(f"\n [INFO] Total Duration: {(response['total_duration']/1e9):.2f} seconds")

PROMPT_2 = 'and of Italy?'
response = ollama.chat(model=MODEL, messages=[
{'role': 'user','content': PROMPT_1,},
{'role': 'assistant','content': resp_1,},
{'role': 'user','content': PROMPT_2,},])
resp_2 = response['message']['content']
print(f"\n{resp_2}")
print(f"\n [INFO] Total Duration: {(response['total_duration']/1e9):.2f} seconds")
```

Nel codice sopra, stiamo eseguendo due query e il secondo prompt considera il risultato del primo.
Ecco come ha risposto il modello:

```
The capital of France is **Paris**. FR

 [INFO] Total Duration: 2.82 seconds

The capital of Italy is **Rome**. IT

 [INFO] Total Duration: 4.46 seconds
```

Ottenere una descrizione dell'immagine:
Allo stesso modo in cui abbiamo utilizzato il modello `LlaVa-PHI-3` con la riga di comando per analizzare un'immagine, lo stesso può essere fatto qui con Python. Usiamo la stessa immagine di Parigi, ma ora con `ollama.generate()`:

```
MODEL = 'llava-phi3:3.8b'
PROMPT = "Describe this picture"

with open('image_test_1.jpg', 'rb') as image_file:
    img = image_file.read()

response = ollama.generate(
    model=MODEL,
    prompt=PROMPT,
    images= [img]
)
print(f"\n{response['response']}")
print(f"\n [INFO] Total Duration: {(res['total_duration']/1e9):.2f} seconds")
```

Ecco il risultato:

```
This image captures the iconic cityscape of Paris, France. The vantage point
is high, providing a panoramic view of the Seine River that meanders through
the heart of the city. Several bridges arch gracefully over the river,
connecting different parts of the city. The Eiffel Tower, an iron lattice
structure with a pointed top and two antennas on its summit, stands tall in the
background, piercing the sky. It is painted in a light gray color, contrasting
against the blue sky speckled with white clouds.

The buildings that line the river are predominantly white or beige, their uniform
color palette broken occasionally by red roofs peeking through. The Seine River
itself appears calm and wide, reflecting the city's architectural beauty in its
surface. On either side of the river, trees add a touch of green to the urban
landscape.

The image is taken from an elevated perspective, looking down on the city. This
viewpoint allows for a comprehensive view of Paris's beautiful architecture and
layout. The relative positions of the buildings, bridges, and other structures
create a harmonious composition that showcases the city's charm.

In summary, this image presents a serene day in Paris, with its architectural
marvels - from the Eiffel Tower to the river-side buildings - all bathed in soft
colors under a clear sky.
```

```
[INFO] Total Duration: 256.45 seconds
```

Il modello ha impiegato circa 4 minuti (256,45 s) per restituire una descrizione dettagliata dell'immagine.

> Nel notebook 10-Ollama_Python_Library, è possibile trovare gli esperimenti con la libreria Python Ollama.

Chiamata di Funzione

Finora, possiamo osservare che utilizzando la risposta del modello in una variabile, possiamo incorporarla efficacemente in progetti reali. Tuttavia, sorge un problema importante quando il modello fornisce risposte diverse allo stesso input. Ad esempio, supponiamo di aver bisogno solo del nome della capitale di un paese e delle sue coordinate come risposta del modello negli esempi precedenti, senza ulteriori informazioni, anche quando si utilizzano modelli dettagliati come Microsoft Phi. Per garantire risposte coerenti, possiamo utilizzare la "chiamata di funzione Ollama", che è completamente compatibile con l'API OpenAI.

Ma cos'è esattamente la "chiamata di funzione"?

Nell'intelligenza artificiale moderna, la chiamata di funzione con Large Language Models (LLM) consente a questi modelli di eseguire azioni che vanno oltre la generazione di testo. Integrandosi con funzioni o API esterne, gli LLM possono accedere a dati in tempo reale, automatizzare attività e interagire con vari sistemi.

Ad esempio, invece di rispondere semplicemente a una query sul meteo, un LLM può chiamare un'API meteo per recuperare le condizioni attuali e fornire informazioni accurate e aggiornate. Questa capacità migliora la pertinenza e l'accuratezza delle risposte del modello e lo rende uno strumento potente per guidare flussi di lavoro e automatizzare i processi, trasformandolo in un partecipante attivo nelle applicazioni del mondo reale.

Per maggiori dettagli sulla "Function Calling" chiamata di funzione, c'è questo video realizzato da Marvin Prison:

https://www.youtube.com/embed/eHfMCtlsb1o

Creiamo un progetto.

Vogliamo creare un'*app* in cui l'utente inserisce il nome di un Paese e ottiene, come output, la distanza in km dalla capitale di tale Paese e la posizione dell'app (per semplicità, useremo Santiago del Cile come posizione dell'app).

Una volta che l'utente inserisce il nome di un paese, il modello restituirà il nome della sua capitale (come stringa) e la latitudine e la longitudine di tale città (in float). Utilizzando queste coordinate, possiamo usare una semplice libreria Python (haversine) per calcolare la distanza tra quei 2 punti.

L'idea di questo progetto è dimostrare una combinazione di interazione del modello linguistico, gestione dei dati strutturati con Pydantic e calcoli geospaziali utilizzando la formula di Haversine (informatica tradizionale).
Per prima cosa, installiamo alcune librerie. Oltre a *Haversine*, la principale è la libreria Python OpenAI, che fornisce un comodo accesso all'API REST OpenAI da qualsiasi applicazione Python 3.7+. L'altra è Pydantic (e instructor), una libreria robusta di convalida dei dati e gestione delle impostazioni progettata da Python per migliorare la robustezza e l'affidabilità della nostra base di codice. In breve, *Pydantic* ci aiuterà a garantire che la risposta del nostro modello sia sempre coerente.

```
pip install haversine
pip install openai
pip install pydantic
pip install instructor
```

Ora, dovremmo creare uno script Python progettato per interagire con il nostro modello (LLM) per determinare le coordinate della capitale di un paese e calcolare la distanza da Santiago del Cile a quella capitale.
Diamo un'occhiata al codice:

1. Importazione delle Librerie

```
import sys
from haversine import haversine
from openai import OpenAI
from pydantic import BaseModel, Field
import instructor
```

- **sys**: Fornisce accesso a parametri e funzioni specifici del sistema. Viene utilizzato per ottenere argomenti dalla riga di comando.
- **haversine**: Una funzione della libreria haversine che calcola la distanza tra due punti geografici utilizzando la formula Haversine.
- **openAI**: Un modulo per interagire con l'API OpenAI (anche se viene utilizzato insieme a una configurazione locale, Ollama). Qui tutto è offline.
- **pydantic**: Fornisce la convalida dei dati e la gestione delle impostazioni utilizzando annotazioni di tipo Python. Viene utilizzato per definire la struttura dei dati di risposta previsti.
- **instructor**: Un modulo viene utilizzato per applicare patch al client OpenAI per funzionare in una modalità specifica (probabilmente correlata alla gestione dei dati strutturati).

2. Definizione di Input e Modello

```
country = sys.argv[1]        # Get the country from command-line arguments
MODEL = 'phi3.5:3.8b'      # The name of the model to be used
mylat = -33.33               # Latitude of Santiago de Chile
mylon = -70.51               # Longitude of Santiago de Chile
```

- **country**: In uno script Python, è possibile ottenere il nome del paese dagli argomenti della riga di comando. In un notebook Jupyter, possiamo immettere il suo nome, ad esempio,
  - `country = "France"`
- **MODEL**: Specifica il modello utilizzato, che è, in questo esempio, phi3.5.
- **mylat e mylon**: Coordinate di Santiago del Cile, utilizzate come punto di partenza per il calcolo della distanza.

3. Definizione della Struttura dei Dati di Risposta

```
class CityCoord(BaseModel):
    city: str = Field(..., description="Name of the city")
```

```
    lat: float = Field(..., description="Decimal Latitude of the city")
    lon: float = Field(..., description="Decimal Longitude of the city")
```

- **CityCoord**: Un modello Pydantic che definisce la struttura prevista della risposta dal LLM. Si aspetta tre campi: city (nome della città), lat (latitudine) e lon (longitudine).

4. Impostazione del Client OpenAI

```
client = instructor.patch(
    OpenAI(
        base_url="http://localhost:11434/v1",  # Local API base URL (Ollama)
        api_key="ollama",                      # API key (not used)
    ),
    mode=instructor.Mode.JSON,                 # Mode for structured JSON output
)
```

- **OpenAI**: Questa configurazione inizializza un client OpenAI con un URL di base locale e una chiave API (ollama). Utilizza un server locale.
- **instructor.patch**: Applica patch al client OpenAI per funzionare in modalità JSON, abilitando un output strutturato che corrisponde al modello Pydantic.

5. Generazione della Risposta

```
resp = client.chat.completions.create(
    model=MODEL,
    messages=[
        {
            "role": "user",
            "content": f"return the decimal latitude and decimal longitude \
            of the capital of the {country}."
        }

    ],
    response_model=CityCoord,
    max_retries=10
)
```

- **client.chat.completions.create**: Chiama l'LLM per generare una risposta.
- **model**: Specifica il modello da utilizzare (llava-phi3).
- **messages**: Contiene il prompt per l'LLM, che chiede latitudine e longitudine della capitale del paese specificato.
- **response_model**: Indica che la risposta deve essere conforme al modello CityCoord.
- **max_retries**: Numero massimo di tentativi di ripetizione se la richiesta fallisce.

6. Calcolo della Distanza

```
distance = haversine((mylat, mylon), (resp.lat, resp.lon), unit='km')
print(f"Santiago de Chile is about {int(round(distance, -1)):,} \
        kilometers away from {resp.city}.")
```

- **haversine**: Calcola la distanza tra Santiago del Cile e la capitale restituita dall'LLM utilizzando le rispettive coordinate.
- **(mylat, mylon)**: Coordinate di Santiago del Cile.
- **resp.city**: Nome della capitale del paese
- **(resp.lat, resp.lon)**: Le coordinate della capitale sono fornite dalla risposta dell'LLM.
- **unit='km'**: Specifica che la distanza deve essere calcolata in chilometri.
- **print**: Restituisce la distanza, arrotondata ai 10 chilometri più vicini, con migliaia di separatori per una migliore leggibilità.

Esecuzione del codice

Se inseriamo paesi diversi, ad esempio Francia, Colombia e Stati Uniti, possiamo notare che riceviamo sempre le stesse informazioni strutturate:

```
Santiago de Chile is about 8,060 kilometers away from Washington, D.C..
Santiago de Chile is about 4,250 kilometers away from Bogotá.
Santiago de Chile is about 11,630 kilometers away from Paris.
```

Eseguendo il codice come script, il risultato verrà stampato sul terminale:

```
mjrovai@rpi-5: ~/Documents/OLLAMA
File  Edit  Tabs  Help
mjrovai@rpi-5:~/Documents/OLLAMA $
mjrovai@rpi-5:~/Documents/OLLAMA $ python calc_distance.py "United States"

Santiago de Chile is about 8,060 kilometers away from Washington, D.C..

mjrovai@rpi-5:~/Documents/OLLAMA $ python calc_distance.py "Colombia"

Santiago de Chile is about 4,250 kilometers away from Bogotá.

mjrovai@rpi-5:~/Documents/OLLAMA $ python calc_distance.py "France"

Santiago de Chile is about 11,630 kilometers away from Paris.

mjrovai@rpi-5:~/Documents/OLLAMA $
```

E i calcoli sono piuttosto buoni!

Nel notebook 20-Ollama_Function_Calling, è possibile trovare esperimenti con tutti i modelli installati.

Aggiunta di immagini

Ora è il momento raccogliere il tutto! Modifichiamo lo script in modo che invece di immettere il nome del paese (come testo), l'utente immetta un'immagine e l'applicazione (basata su SLM) restituisca la città nell'immagine e la sua posizione geografica. Con quei dati, possiamo calcolare la distanza come prima.

Per semplicità, implementeremo questo nuovo codice in due passaggi. Innanzitutto, LLM analizzerà l'immagine e creerà una descrizione (testo). Questo testo verrà passato a un'altra istanza, dove il modello estrarrà le informazioni necessarie per il passaggio.
Inizieremo ad importare le librerie

```
import sys
import time
from haversine import haversine
import ollama
from openai import OpenAI
from pydantic import BaseModel, Field
import instructor
```

Possiamo vedere l'immagine se si esegue il codice su Jupyter Notebook. Per questo dobbiamo anche importare:

```
import matplotlib.pyplot as plt
from PIL import Image
```

Tali librerie non sono necessarie se eseguiamo il codice come uno script.

Ora, definiamo il modello e le coordinate locali:

```
MODEL = 'llava-phi3:3.8b'
mylat = -33.33
mylon = -70.51
```

Possiamo effettuare il download di una nuova immagine, ad esempio Machu Picchu da Wikipedia. Sul Notebook possiamo vederla:

```
# Load the image
img_path = "image_test_3.jpg"
img = Image.open(img_path)

# Display the image
plt.figure(figsize=(8, 8))
plt.imshow(img)
plt.axis('off')
#plt.title("Image")
plt.show()
```

Ora, definiamo una funzione che riceverà l'immagine e che `return the decimal latitude and decimal longitude of the city in the image, its name, and what country it is located` [restituirà la latitudine decimale e la longitudine decimale della città nell'immagine, il suo nome e il paese in cui si trova].

```
def image_description(img_path):
    with open(img_path, 'rb') as file:
        response = ollama.chat(
            model=MODEL,
            messages=[
              {
                'role': 'user',
                'content': '''return the decimal latitude and decimal longitude
                              of the city in the image, its name, and
                              what country it is located''',
                'images': [file.read()],
              },
            ],
            options = {
```

```
                'temperature': 0,
                }

        )

    #print(response['message']['content'])
    return response['message']['content']
```

Possiamo stampare l'intera risposta per debug.

La descrizione dell'immagine generata per la funzione verrà passata di nuovo come prompt per il modello.

```
start_time = time.perf_counter()  # Start timing

class CityCoord(BaseModel):
    city: str = Field(..., description="Name of the city in the image")
    country: str = Field(..., description="""Name of the country where"
                                            the city in the image is located
                                            """)
    lat: float = Field(..., description="""Decimal Latitude of the city in"
                                           the image""")
    lon: float = Field(..., description="""Decimal Longitude of the city in"
                                          the image""")

# enables `response_model` in create call
client = instructor.patch(
    OpenAI(
        base_url="http://localhost:11434/v1",
        api_key="ollama"
    ),
    mode=instructor.Mode.JSON,
)

image_description = image_description(img_path)
# Send this description to the model
resp = client.chat.completions.create(
    model=MODEL,
    messages=[
        {
            "role": "user",
            "content": image_description,
        }

    ],
    response_model=CityCoord,
    max_retries=10,
    temperature=0,
)
```

Se stampiamo la descrizione dell'immagine, otterremo:

```
The image shows the ancient city of Machu Picchu, located in Peru. The city is
perched on a steep hillside and consists of various structures made of stone. It
is surrounded by lush greenery and towering mountains. The sky above is blue with
scattered clouds.

Machu Picchu's latitude is approximately 13.5086° S, and its longitude is around
72.5494° W.
```

E la seconda risposta del modello (`resp`) sarà:

```
CityCoord(city='Machu Picchu', country='Peru', lat=-13.5086, lon=-72.5494)
```

Ora possiamo effettuare un "Post-Processing", calcolando la distanza e preparando la risposta finale:

```
distance = haversine((mylat, mylon), (resp.lat, resp.lon), unit='km')

print(f"\n The image shows {resp.city}, with lat:{round(resp.lat, 2)} and \
      long: {round(resp.lon, 2)}, located in {resp.country} and about \
            {int(round(distance, -1)):,} kilometers away from \
            Santiago, Chile.\n")

end_time = time.perf_counter()  # End timing
```

```
elapsed_time = end_time - start_time  # Calculate elapsed time
print(f" [INFO] ==> The code (running {MODEL}), took {elapsed_time:.1f} \
      seconds to execute.\n")
```

E otterremo:

```
The image shows Machu Picchu, with lat:-13.16 and long: -72.54, located in Peru
 and about 2,250 kilometers away from Santiago, Chile.

 [INFO] ==> The code (running llava-phi3:3.8b), took 491.3 seconds to execute.
```

Nel notebook 30-Function_Calling_with_images è possibile trovare gli esperimenti con più immagini.
Scarichiamo ora lo script `calc_distance_image.py` da **Errore. Riferimento a collegamento ipertestuale non valido.** ed eseguiamolo sul terminale con il comando:

```
python calc_distance_image.py /home/mjrovai/Documents/OLLAMA/image_test_3.jpg
```

Immettere la patch completa dell'immagine di Machu Picchu come argomento. Otterremo lo stesso risultato precedente.

Per wuanto riguarda Parigi?

Naturalmente, ci sono molti modi per ottimizzare il codice utilizzato qui. Tuttavia, l'idea è di esplorare il notevole potenziale della *function calling* con SLM nell'edge, consentendo a tali modelli di integrarsi con funzioni o API esterne. Andando oltre la generazione di testo, gli SLM possono accedere a dati in tempo reale, automatizzare attività e interagire con vari sistemi.

SLM: Tecniche di Ottimizzazione

I Large Language Model (LLM) hanno rivoluzionato l'elaborazione del linguaggio naturale, ma la loro distribuzione e ottimizzazione presentano sfide uniche. Un problema significativo è la tendenza degli LLM (e più in particolare degli SLM) a generare informazioni che sembrano plausibili ma di fatto errate, un fenomeno noto come **allucinazione**. Ciò si verifica quando i modelli producono contenuti che sembrano coerenti ma non sono basati sulla verità o sui fatti del mondo reale.
Altre sfide includono le immense risorse computazionali richieste per l'addestramento e l'esecuzione di questi modelli, la difficoltà nel mantenere aggiornate le conoscenze all'interno del modello e la necessità di adattamenti specifici per dominio. Problemi di privacy sorgono anche quando si gestiscono dati sensibili durante l'addestramento o l'inferenza. Inoltre, garantire prestazioni coerenti in diverse attività e mantenere un uso etico di questi potenti strumenti presentano sfide continue. Affrontare questi problemi è

fondamentale per l'implementazione efficace e responsabile di LLM in applicazioni del mondo reale.
Le tecniche fondamentali per migliorare le prestazioni e l'efficienza di LLM (e SLM) sono Fine-tuning, Prompt engineering e Retrieval-Augmented Generation (RAG).

- **Fine-tuning**, sebbene richieda più risorse, offre un modo per specializzare LLM per domini o attività particolari. Questo processo comporta un ulteriore addestramento del modello su set di dati attentamente curati, consentendogli di adattare la sua vasta conoscenza generale ad applicazioni specifiche. Il Fine-tuning può portare a miglioramenti sostanziali nelle prestazioni, specialmente in campi specializzati o per casi d'uso unici.
- **Prompt engineering** è in prima linea nell'ottimizzazione LLM. Creando attentamente prompt di input, possiamo guidare i modelli per produrre output più accurati e pertinenti. Questa tecnica comporta la strutturazione di query che sfruttano le conoscenze e le capacità pre-addestrate del modello, spesso incorporando esempi o istruzioni specifiche per modellare la risposta desiderata.
- **Retrieval-Augmented Generation (RAG)** rappresenta un altro approccio potente per migliorare le prestazioni LLM. Questo metodo combina la vasta conoscenza incorporata nei modelli pre-addestrati con la capacità di accedere e incorporare informazioni esterne aggiornate. Recuperando dati pertinenti per integrare il processo decisionale del modello, RAG può migliorare significativamente l'accuratezza e ridurre la probabilità di generare informazioni obsolete o false.

Per le applicazioni edge, è più utile concentrarsi su tecniche come RAG che possono migliorare le prestazioni del modello senza dover effettuare un "fine-tuning" sul dispositivo. Esploriamolo.

Implementazione del RAG

In un'interazione di base tra un utente e un modello linguistico, l'utente pone una domanda, che viene inviata come prompt al modello. Il modello genera una risposta basata esclusivamente sulla sua conoscenza pre-addestrata. In un processo RAG, c'è un passaggio aggiuntivo tra la domanda dell'utente e la risposta del modello. La domanda dell'utente innesca un processo di recupero da una "knowledge base".

Un semplice progetto RAG

Ecco i passaggi per implementare una Retrieval Augmented Generation (RAG) di base:

- **Determinare il tipo di documenti che si utilizzeranno:** I tipi migliori sono documenti da cui possiamo ottenere testo pulito e non oscurato. I PDF possono essere

problematici perché sono progettati per la stampa, non per estrarre testo sensato. Per lavorare con i PDF, dovremmo ottenere il documento di origine o utilizzare strumenti per gestirli.

- **Suddividere il testo in blocchi:** Non possiamo archiviare il testo come un unico lungo flusso a causa delle limitazioni delle dimensioni del contesto e della potenziale confusione. La suddivisione in blocchi comporta la suddivisione del testo in parti più piccole. Il testo in blocchi ha molti modi, come conteggio dei caratteri, token, parole, paragrafi o sezioni. È anche possibile sovrapporre i blocchi.
- **Crea embedding:** Gli "embedding" [incorporamenti] sono rappresentazioni numeriche del testo che catturano il significato semantico. Creiamo incorporamenti passando ogni blocco di testo attraverso un particolare modello di embedding. Il modello genera un vettore, la cui lunghezza dipende dal modello di embedding utilizzato. Dovremmo estrarre uno (o più) modelli di embedding da Ollama, per eseguire questa attività. Ecco alcuni esempi di modelli di embedding disponibili su Ollama.

Modello	Dimensione del parametro	Dimensione dell'Embedding
mxbai-embed-large	334M	1024
nomic-embed-text	137M	768
all-minilm	23M	384

  In genere, dimensioni dell'embedding maggiori catturano informazioni più sfumate sull'input. Tuttavia, richiedono anche più risorse per l'elaborazione e un numero maggiore di parametri dovrebbe aumentare la latenza (ma anche la qualità della risposta).

- **Memorizzare i blocchi e gli embedding in un database vettoriale:** Avremo bisogno di un modo per trovare in modo efficiente i blocchi di testo più rilevanti per un dato prompt, ed è qui che entra in gioco un database vettoriale. Useremo Chromadb, un database vettoriale open source nativo IA, che semplifica la creazione di RAG creando conoscenze, fatti e competenze collegabili per LLM. Vengono memorizzati sia embedding che il testo sorgente per ogni blocco.
- **Creare il prompt:** Quando abbiamo una domanda, creiamo un embedding e interroghiamo il database vettoriale per i blocchi più simili. Poi, selezioniamo i primi risultati e includiamo il loro testo nel prompt.

L'obiettivo di RAG è fornire al modello le informazioni più rilevanti dai nostri documenti, consentendogli di generare risposte più accurate e informative. Quindi, implementiamo un semplice esempio di un SLM che incorpora un set particolare di fatti sulle api ("Bee Facts"). All'interno dell'ambiente `ollama`, si inserisce il comando nel terminale per l'installazione di Chromadb:

```
pip install ollama chromadb
```

Tiriamo fuori un modello di embedding intermedio, `nomic-embed-text`

```
ollama pull nomic-embed-text
```

E creiamo una directory di lavoro:

```
cd Documents/OLLAMA/
mkdir RAG-simple-bee
cd RAG-simple-bee/
```

Creiamo un nuovo notebook Jupyter, 40-RAG-simple-bee per qualche esplorazione:
Importare le librerie necessarie:

```
import ollama
import chromadb
import time
```

E definiamo i modelli aor:

```
EMB_MODEL = "nomic-embed-text"
MODEL = 'llama3.2:3B'
```

Inizialmente, dovrebbe essere creata una “knowledge base” sui fatti sulle api. Ciò comporta la raccolta di documenti rilevanti e la loro conversione in embedding vettoriali. Questi embedding vengono poi archiviati in un database vettoriale, consentendo in seguito ricerche di similarità efficienti. Si immette il “document”, una base di “fatti sulle api” come un elenco:

```
documents = [
    "Bee-keeping, also known as apiculture, involves the maintenance of bee \
    colonies, typically in hives, by humans.",
    "The most commonly kept species of bees is the European honey bee (Apis \
    mellifera).",

    ...

    "There are another 20,000 different bee species in the world.",
    "Brazil alone has more than 300 different bee species, and the \
    vast majority, unlike western honey bees, don’t sting.",
    "Reports written in 1577 by Hans Staden, mention three native bees \
    used by indigenous people in Brazil.",
    "The indigenous people in Brazil used bees for medicine and food purposes",
    "From Hans Staden report: probable species: mandaçaia (Melipona \
    quadrifasciata), mandaguari (Scaptotrigona postica) and jataí-amarela \
    (Tetragonisca angustula)."
]
```

Non abbiamo bisogno di “suddividere” il documento qui perché useremo ogni elemento dell’elenco e un blocco.

Ora creeremo il nostro database di embedding vettoriale `bee_facts` e memorizzeremo il documento in esso:

```
client = chromadb.Client()
collection = client.create_collection(name="bee_facts")

# store each document in a vector embedding database
for i, d in enumerate(documents):
  response = ollama.embeddings(model=EMB_MODEL, prompt=d)
  embedding = response["embedding"]
  collection.add(
    ids=[str(i)],
    embeddings=[embedding],
    documents=[d]
  )
```

ra che abbiamo creato la nostra “Knowledge Base”, possiamo iniziare a fare query, recuperando dati da essa:

Query utente: Il processo inizia quando un utente pone una domanda, come "Quante api ci sono in una colonia? Chi depone le uova e in che quantità? E per quanto riguarda parassiti e malattie comuni?"

```
prompt = "How many bees are in a colony? Who lays eggs and how much? How about\
          common pests and diseases?"
```

Query Embedding: La domanda dell'utente viene convertita in un embedding vettoriale utilizzando **lo stesso modello di embedding** utilizzato per la knowledge base.

```
response = ollama.embeddings(
  prompt=prompt,
  model=EMB_MODEL
)
```

Recupero di Documenti Pertinenti: Il sistema esegue una ricerca nella knowledge base utilizzando il "query embedding" per trovare i documenti più pertinenti (in questo caso, i 5 più probabili). Ciò avviene tramite una ricerca di similarità, che confronta la "query embedding" con gli embedding di documenti nel database.

```
results = collection.query(
  query_embeddings=[response["embedding"]],
  n_results=5
)
data = results['documents']
```

Prompt Augmentation: Le informazioni rilevanti recuperate vengono combinate con la query utente originale per creare un prompt "aumentato". Questo prompt ora contiene la domanda dell'utente e i fatti pertinenti dalla knowledge base.

```
prompt=f"Using this data: {data}. Respond to this prompt: {prompt}",
```

Generazione di Risposte: Il prompt aumentato viene poi immesso in un modello linguistico, in questo caso il modello `llama3.2:3b`. Il modello utilizza questo contesto arricchito per generare una risposta completa. Parametri come temperatura, top_k e top_p vengono impostati per controllare la casualità e la qualità della risposta generata.

```
output = ollama.generate(
  model=MODEL,
  prompt=f"Using this data: {data}. Respond to this prompt: {prompt}",

  options={
    "temperature": 0.0,
    "top_k":10,
    "top_p":0.5                          }
)
```

Response Delivery: Infine, il sistema restituisce all'utente la risposta generata.

```
print(output['response'])
Based on the provided data, here are the answers to your questions:

1. How many bees are in a colony?
A typical bee colony can contain between 20,000 and 80,000 bees.

2. Who lays eggs and how much?
The queen bee lays up to 2,000 eggs per day during peak seasons.

3. What about common pests and diseases?
Common pests and diseases that affect bees include varroa mites, hive beetles,
and foulbrood.
```

Creiamo una funzione che ci aiuti a rispondere a nuove domande:

```
def rag_bees(prompt, n_results=5, temp=0.0, top_k=10, top_p=0.5):
    start_time = time.perf_counter()  # Start timing

    # generate an embedding for the prompt and retrieve the data
    response = ollama.embeddings(
      prompt=prompt,
      model=EMB_MODEL
    )


    results = collection.query(
```

```
      query_embeddings=[response["embedding"]],
      n_results=n_results
    )

    data = results['documents']

    # generate a response combining the prompt and data retrieved
    output = ollama.generate(
      model=MODEL,
      prompt=f"Using this data: {data}. Respond to this prompt: {prompt}",

      options={
        "temperature": temp,
        "top_k": top_k,
        "top_p": top_p                          }
    )


    print(output['response'])


    end_time = time.perf_counter()  # End timing
    elapsed_time = round((end_time - start_time), 1)  # Calculate elapsed time

    print(f"\n [INFO] ==> The code for model: {MODEL}, took {elapsed_time}s \
          to generate the answer.\n")
```

Ora possiamo creare delle query e chiamare la funzione:

```
prompt = "Are bees in Brazil?"
rag_bees(prompt)
Yes, bees are found in Brazil. According to the data, Brazil has more than 300
different bee species, and indigenous people in Brazil used bees for medicine and
food purposes. Additionally, reports from 1577 mention three native bees used by
indigenous people in Brazil.

 [INFO] ==> The code for model: llama3.2:3b, took 22.7s to generate the answer.
```

A proposito, se il modello utilizzato supporta più lingue, possiamo utilizzarlo (ad esempio, il portoghese), anche se il set di dati è stato creato in inglese:

```
prompt = "Existem abelhas no Brazil?"
rag_bees(prompt)
Sim, existem abelhas no Brasil! De acordo com o relato de Hans Staden, há três
espécies de abelhas nativas do Brasil que foram mencionadas: mandaçaia (Melipona
quadrifasciata), mandaguari (Scaptotrigona postica) e jataí-amarela (Tetragonisca
angustula). Além disso, o Brasil é conhecido por ter mais de 300 espécies diferentes de a
belhas, a maioria das quais não é agressiva e não põe veneno.

 [INFO] ==> The code for model: llama3.2:3b, took 54.6s to generate the answer.
```

Andando Oltre

I piccoli modelli LLM testati hanno funzionato bene sull'edge, sia nel testo che con le immagini, ma ovviamente avevano un'elevata latenza per quanto riguarda quest'ultima. Una combinazione di modelli specifici e dedicati può portare a risultati migliori; ad esempio, in casi reali, un modello di rilevamento degli oggetti (come YOLO) può ottenere una descrizione generale e un conteggio degli oggetti su un'immagine che, una volta passati a un LLM, possono aiutare a estrarre informazioni e azioni essenziali.
Secondo Avi Baum, CTO di Hailo,

> Nel vasto panorama dell'intelligenza artificiale (IA), uno dei viaggi più intriganti è stata l'evoluzione dell'IA nell'edge. Questo viaggio ci ha portato dalla classica visione artificiale ai regni dell'IA discriminativa, dell'IA potenziata e ora, alla rivoluzionaria frontiera dell'IA generativa. Ogni passo ci ha avvicinato a un futuro in cui i sistemi intelligenti si integrano perfettamente con la nostra vita quotidiana, offrendo un'esperienza immersiva non solo di percezione ma anche di creazione nel palmo della nostra mano.

Conclusione

Questo laboratorio ha dimostrato come un Raspberry Pi 5 può essere trasformato in un potente hub AI in grado di eseguire "large language model (LLM)" per analisi e approfondimenti dei dati in tempo reale e in loco utilizzando Ollama e Python. La versatilità e la potenza del Raspberry Pi, unite alle capacità di LLM leggeri come Llama 3.2 e LLaVa-Phi-3-mini, lo rendono un'eccellente piattaforma per applicazioni di edge computing.
Il potenziale di esecuzione di LLM sull'edge si estende ben oltre la semplice elaborazione dei dati, come negli esempi di questo laboratorio. Ecco alcuni suggerimenti innovativi per l'utilizzo di questo progetto:

1. Smart Home Automation:

- Integrare gli SLM per interpretare i comandi vocali o analizzare i dati dei sensori per l'automazione domestica intelligente. Ciò potrebbe includere il monitoraggio e il controllo in tempo reale di dispositivi domestici, sistemi di sicurezza e gestione energetica, tutti elaborati localmente senza fare affidamento sui servizi cloud.

2. Raccolta e Analisi dei Dati sul Campo:

- Distribuire gli SLM su Raspberry Pi in configurazioni remote o mobili per la raccolta e l'analisi dei dati in tempo reale. Può essere utilizzato in agricoltura per monitorare la salute delle colture, negli studi ambientali per il monitoraggio della fauna selvatica o nella risposta ai disastri per la consapevolezza della situazione e la gestione delle risorse.

3. Strumenti Didattici:

- Creare strumenti didattici interattivi che sfruttano gli SLM per fornire feedback immediato, traduzione linguistica e tutoraggio. Può essere particolarmente utile nelle regioni in via di sviluppo con accesso limitato a tecnologie avanzate e connettività Internet.

4. Applicazioni Sanitarie:

- Utilizzare gli SLM per la diagnosi medica e il monitoraggio dei pazienti. Possono fornire analisi in tempo reale dei sintomi e suggerire potenziali trattamenti. Può essere integrato in piattaforme di telemedicina o dispositivi sanitari portatili.

5. Intelligence Aziendale Locale:

- Implementare gli SLM in ambienti di vendita al dettaglio o di piccole imprese per analizzare il comportamento dei clienti, gestire l'inventario e ottimizzare le operazioni. La capacità di elaborare i dati localmente garantisce la privacy e riduce la dipendenza dai servizi esterni.

6. IoT Industriale:

- Integrare gli SLM nei sistemi IoT industriali per manutenzione predittiva, controllo qualità e ottimizzazione dei processi. Raspberry Pi può fungere da unità di elaborazione dati localizzata, riducendo la latenza e migliorando l'affidabilità dei sistemi automatizzati.

7. Veicoli Autonomi:

- Utilizzare gli SLM per elaborare dati sensoriali da veicoli autonomi, consentendo decisioni e navigazione in tempo reale. Questo può essere applicato a droni, robot e auto a guida autonoma per una maggiore autonomia e sicurezza.

8. Patrimonio Culturale e Turismo:

- Implementare gli SLM per fornire siti del patrimonio culturale e guide museali interattive e informative. I visitatori possono utilizzare questi sistemi per ottenere informazioni e approfondimenti in tempo reale, migliorando la loro esperienza senza connettività Internet.

9. Progetti Artistici e Creativi:

- Utilizzare SLM per analizzare e generare contenuti creativi, come musica, arte e letteratura. Questo può promuovere progetti innovativi nei settori creativi e consentire esperienze interattive uniche in mostre e spettacoli.

10. Tecnologie Assistenziali Personalizzate:

- Sviluppare tecnologie assistenziali per persone con disabilità, fornendo supporto personalizzato e adattivo tramite testo in tempo reale, traduzione di lingue e altri strumenti accessibili.

Risorse

- 10-Ollama_Python_Library notebook
- 20-Ollama_Function_Calling notebook
- 30-Function_Calling_with_images notebook
- 40-RAG-simple-bee notebook
- calc_distance_image python script

Lab Condivisi

I lab in questa sezione coprono argomenti e tecniche applicabili a diverse piattaforme hardware. Questi laboratori sono progettati per essere indipendenti da schede specifiche, consentendo di concentrarsi sui concetti fondamentali e sugli algoritmi utilizzati nelle applicazioni (tiny) ML.
Esplorando questi “shared lab”, si otterrà una comprensione più approfondita delle sfide e delle soluzioni comuni nell’apprendimento automatico embedded. Le conoscenze e le competenze acquisite qui saranno preziose indipendentemente dall’hardware specifico con cui lavorerà in futuro.

Esercizio	Nicla Vision	XIAO ESP32S3
KWS Feature Engineering	✓ Link	✓ Link
Blocco delle Feature Spettrali DSP	✓ Link	✓ Link

KWS Feature Engineering

DALL·E 3 Prompt: Scena di un cartone animato in stile anni '50 ambientata in una sala di ricerca Due scienziati, uno con una lente di ingrandimento e l'altro che prende appunti, esaminano grandi grafici appesi al muro. Questi grafici raffigurano FFT e curve temporali relative all'analisi dei dati audio. La sala ha un'atmosfera retrò, con tavoli in legno, lampade vintage e classici strumenti di analisi audio.

Introduzione

In questo tutorial pratico, l'enfasi è posta sul ruolo critico che la "feature engineering" [ingegneria delle funzionalità] svolge nell'ottimizzazione delle prestazioni dei modelli di machine learning applicati alle attività di classificazione audio, come il riconoscimento vocale. È essenziale essere consapevoli che le prestazioni di qualsiasi modello di apprendimento automatico dipendono in larga misura dalla qualità delle feature utilizzate e ci occuperemo della meccanica "sotto il cofano" dell'estrazione delle feature, concentrandoci principalmente sui Mel-frequency Cepstral Coefficient (MFCC), una pietra miliare nel campo dell'elaborazione del segnale audio.

I modelli di apprendimento automatico, in particolare gli algoritmi tradizionali, non comprendono le onde audio. Comprendono i numeri disposti in modo significativo, ovvero le feature. Queste feature incapsulano le caratteristiche del segnale audio, rendendo più facile per i modelli distinguere tra suoni diversi.

Questo tutorial si occuperà della generazione di feature specificamente per la classificazione audio. Ciò può essere particolarmente interessante per l'applicazione di machine learning a una varietà di dati audio, sia per il riconoscimento vocale, la categorizzazione musicale, la classificazione degli insetti basata sui suoni del battito delle ali o altre attività di analisi del suono

Il KWS

L'applicazione TinyML più comune è Keyword Spotting (KWS), un sottoinsieme del campo più ampio del riconoscimento vocale. Mentre il riconoscimento vocale generale trascrive tutte le parole pronunciate in testo, il Keyword Spotting si concentra sul rilevamento di "parole chiave" o "wake word" specifiche in un flusso audio continuo. Il sistema è addestrato a riconoscere queste parole chiave come frasi o parole predefinite, come *yes* o *no*. In breve, KWS è una forma specializzata di riconoscimento vocale con il suo set di sfide e requisiti. Ecco un tipico processo KWS che utilizza MFCC Feature Converter:

Applicazioni di KWS

- **Assistenti Vocali**: In dispositivi come Alexa di Amazon o Google Home, KWS viene utilizzato per rilevare la wake word ("Alexa" o "Hey Google") per attivare il dispositivo.
- **Comandi Attivati Tramite Voce**: In contesti automobilistici o industriali, KWS può essere utilizzato per avviare comandi specifici come "Avvia motore" o "Spegni luci".
- **Sistemi di Sicurezza**: I sistemi di sicurezza attivati tramite voce possono utilizzare KWS per autenticare gli utenti in base a una passphrase pronunciata.
- **Servizi di Telecomunicazione**: Le linee del servizio clienti possono utilizzare KWS per instradare le chiamate in base a parole chiave pronunciate.

Differenze dal Riconoscimento Vocale Generale

- **Efficienza Computazionale**: KWS è solitamente progettato per essere meno intensivo dal punto di vista computazionale rispetto al riconoscimento vocale completo, poiché deve riconoscere solo un piccolo set di frasi.
- **Elaborazione in Tempo Reale**: KWS spesso funziona in tempo reale ed è ottimizzato per il rilevamento a bassa latenza delle parole chiave.
- **Vincoli di Risorse**: I modelli KWS sono spesso progettati per essere leggeri, in modo da poter essere eseguiti su dispositivi con risorse computazionali limitate, come microcontrollori o telefoni cellulari.

- **Attività Mirata**: Mentre i modelli di riconoscimento vocale generali sono addestrati per gestire un'ampia gamma di vocabolario e accenti, i modelli KWS sono ottimizzati per riconoscere parole chiave specifiche, spesso in modo accurato in ambienti rumorosi.

Introduzione ai Segnali Audio

Comprendere le proprietà di base dei segnali audio è fondamentale per un'estrazione efficace delle feature caratteristiche e, in ultima analisi, per applicare con successo gli algoritmi di apprendimento automatico nelle attività di classificazione audio. I segnali audio sono forme d'onda complesse che catturano le fluttuazioni della pressione dell'aria nel tempo. Questi segnali possono essere caratterizzati da diversi attributi fondamentali: frequenza di campionamento, frequenza e ampiezza.

- **Frequenza e Ampiezza**: La Frequenza si riferisce al numero di oscillazioni che una forma d'onda subisce per unità di tempo e si misura anche in Hz. Nel contesto dei segnali audio, frequenze diverse corrispondono a pitch [toni] diversi. L'Ampiezza, d'altra parte, misura la grandezza delle oscillazioni e si correla con l'intensità del suono. Sia la frequenza che l'ampiezza sono feature essenziali che catturano le qualità tonali e ritmiche dei segnali audio.
- **Frequenza di Campionamento**: La frequenza di campionamento, spesso indicata in Hertz (Hz), definisce il numero di campioni presi al secondo durante la digitalizzazione di un segnale analogico. Una frequenza di campionamento più elevata consente una rappresentazione digitale più accurata del segnale, ma richiede anche più risorse di elaborazione. Le frequenze di campionamento tipiche includono 44,1 kHz per l'audio di qualità CD e 16 kHz o 8 kHz per le attività di riconoscimento vocale. Comprendere i compromessi nella selezione di una frequenza di campionamento appropriata è essenziale per bilanciare accuratezza ed efficienza computazionale. In generale, con i progetti TinyML, lavoriamo con 16 KHz. Sebbene i toni musicali possano essere uditi a frequenze fino a 20 kHz, la voce raggiunge il massimo a 8 kHz. I sistemi telefonici tradizionali utilizzano una frequenza di campionamento di 8 kHz.

  Per una rappresentazione accurata del segnale, la frequenza di campionamento deve essere almeno il doppio della frequenza più alta presente nel segnale.

- **Dominio del Tempo e Dominio della Frequenza**: I segnali audio possono essere analizzati nei domini del tempo e della frequenza. Nel dominio del tempo, un segnale è rappresentato come una forma d'onda in cui l'ampiezza è tracciata in funzione del tempo. Questa rappresentazione aiuta a osservare feature temporali come inizio e durata, ma le caratteristiche tonali del segnale non sono ben evidenziate. Al contrario, una rappresentazione del dominio della frequenza fornisce una vista delle frequenze costituenti del segnale e delle rispettive ampiezze, in genere ottenute tramite una trasformata di Fourier. Questo è prezioso per le attività che richiedono la comprensione del contenuto spettrale del segnale, come l'identificazione di note musicali o fonemi vocali (il nostro caso).

L'immagine seguente mostra le parole `YES` e `NO` con rappresentazioni tipiche nei domini del tempo (audio grezzo) e della frequenza:

Perché l'Audio Grezzo No?

Sebbene l'utilizzo diretto di dati audio grezzi per attività di apprendimento automatico possa sembrare allettante, questo approccio presenta diverse sfide che lo rendono meno adatto alla creazione di modelli solidi ed efficienti.

L'utilizzo di dati audio grezzi per il Keyword Spotting (KWS), ad esempio, su dispositivi TinyML pone delle sfide dovute alla sua elevata dimensionalità (utilizzando una frequenza di campionamento di 16 kHz), alla complessità computazionale per l'acquisizione di feature temporali, alla suscettibilità al rumore e alla mancanza di feature semanticamente significative, rendendo le tecniche di estrazione di caratteristiche come MFCC una scelta più pratica per applicazioni con risorse limitate.

Ecco alcuni dettagli aggiuntivi sui problemi critici associati all'utilizzo di audio grezzi:

- **Alta Dimensionalità**: I segnali audio, in particolare quelli campionati ad alte velocità, generano grandi quantità di dati. Ad esempio, una clip audio di 1 secondo campionata a 16 kHz avrà 16.000 punti singoli. I dati ad alta dimensionalità aumentano la complessità computazionale, portando a tempi di training più lunghi e costi computazionali più elevati, rendendoli poco pratici per ambienti con risorse limitate. Inoltre, l'ampia gamma dinamica dei segnali audio richiede una quantità significativa di bit per campione, mentre trasmette poche informazioni utili.
- **Dipendenze Temporali**: I segnali audio grezzi hanno strutture temporali che i semplici modelli di apprendimento automatico potrebbero trovare difficili da catturare. Sebbene le reti neurali ricorrenti come LSTM possano modellare tali dipendenze, sono computazionalmente intensive e difficili da addestrare su dispositivi di piccole dimensioni.
- **Rumore e Variabilità**: I segnali audio grezzi spesso contengono rumore di fondo e altri elementi non essenziali che influenzano le prestazioni del modello. Inoltre, lo stesso suono può avere caratteristiche diverse in base a vari fattori come la distanza dal microfono, l'orientamento della sorgente sonora e le proprietà acustiche dell'ambiente, aggiungendo complessità ai dati.
- **Mancanza di Significato Semantico**: L'audio grezzo non contiene intrinsecamente feature semanticamente significative per le attività di classificazione. Feature come

tono, tempo e feature spettrali, che possono essere cruciali per il riconoscimento vocale, non sono direttamente accessibili dai dati della forma d'onda grezza.

- **Ridondanza del Segnale**: I segnali audio spesso contengono informazioni ridondanti, con alcune porzioni del segnale che contribuiscono poco o per niente al compito da svolgere. Questa ridondanza può rendere l'apprendimento inefficiente e potenzialmente portare a un overfitting.

Per queste ragioni, tecniche di estrazione di feature come i Mel-frequency Cepstral Coefficients (MFCCs), le Mel-Frequency Energies (MFEs) e gli spettrogrammi semplici sono comunemente utilizzate per trasformare i dati audio grezzi in un formato più gestibile e informativo. Queste feature catturano le caratteristiche essenziali del segnale audio riducendo dimensionalità e rumore, facilitando un apprendimento automatico più efficace.

Introduzione agli MFCC

Cosa sono gli MFCC?

I Mel-frequency Cepstral Coefficients (MFCC) sono un set di feature derivate dal contenuto spettrale di un segnale audio. Si basano sulle percezioni uditive umane e sono comunemente utilizzati per catturare le feature fonetiche di un segnale audio. Gli MFCC vengono calcolati tramite un processo in più fasi che include pre-enfasi, inquadratura, windowing, applicazione della Fast Fourier Transform (FFT) per convertire il segnale nel dominio della frequenza e, infine, applicazione della Discrete Cosine Transform (DCT). Il risultato è una rappresentazione compatta delle caratteristiche spettrali del segnale audio originale. L'immagine seguente mostra le parole `YES` e `NO` nella loro rappresentazione MFCC:

Questo video spiega i Mel Frequency Cepstral Coefficients (MFCC) e come calcolarli.

Perché gli MFCC sono importanti?

Gli MFCC sono fondamentali per diversi motivi, in particolare nel contesto di Keyword Spotting (KWS) e TinyML:

- **Riduzione della Dimensionalità**: Gli MFCC catturano le caratteristiche spettrali essenziali del segnale audio riducendo significativamente la dimensionalità dei dati, rendendoli ideali per applicazioni TinyML con risorse limitate.
- **Robustezza**: Gli MFCC sono meno sensibili al rumore e alle variazioni di tono e ampiezza, offrendo un set di funzionalità più stabile e robusto per le attività di classificazione audio.
- **Modellazione del Sistema Uditivo Umano**: La scala Mel negli MFCC approssima la risposta dell'orecchio umano a diverse frequenze, rendendoli pratici per il riconoscimento vocale in cui è desiderata una percezione simile a quella umana.

- **Efficienza Computazionale**: Il processo di calcolo degli MFCC è efficiente dal punto di vista computazionale, rendendolo adatto per applicazioni in tempo reale su hardware con risorse computazionali limitate.

In sintesi, gli MFCC offrono un equilibrio tra ricchezza di informazioni ed efficienza computazionale, rendendoli popolari per le attività di classificazione audio, in particolare in ambienti limitati come TinyML.

Calcolo degli MFCC

Il calcolo dei Mel-frequency Cepstral Coefficients (MFCCs) comporta diversi passaggi chiave. Esaminiamoli, che sono particolarmente importanti per le attività di Keyword Spotting (KWS) sui dispositivi TinyML.

- **Pre-enfasi**: Il primo passaggio è la pre-enfasi, che viene applicata per accentuare le componenti ad alta frequenza del segnale audio e bilanciare lo spettro di frequenza. Ciò si ottiene applicando un filtro che amplifica la differenza tra campioni consecutivi. La formula per la pre-enfasi è: $y(t) = x(t) - \alpha\, x(t-1)$, dove α è il fattore di pre-enfasi, in genere intorno a 0.97.
- **Framing**: I segnali audio sono divisi in frame brevi (la *frame length*), in genere da 20 a 40 millisecondi. Ciò si basa sul presupposto che le frequenze in un segnale siano stazionarie per un breve periodo. Il framing aiuta ad analizzare il segnale in slot di tempo così piccoli. Il *frame stride* (o step) sposterà un frame e quello adiacente. Questi step potrebbero essere sequenziali o sovrapposti.
- **Windowing**: Ogni frame viene quindi sottoposto a windowing per ridurre al minimo le discontinuità ai confini del frame. Una funzione window comunemente utilizzata è la finestra di Hamming. Il windowing prepara il segnale per una trasformata di Fourier riducendo al minimo gli effetti alle estremità. L'immagine sottostante mostra tre frame (10, 20 e 30) e i campioni di tempo dopo il windowing (notare che la lunghezza del frame e il frame stride sono di 20 ms):

- **Fast Fourier Transform (FFT)** La Fast Fourier Transform (FFT) viene applicata a ciascun frame sottoposto a windowing per convertirlo dal dominio del tempo al dominio della frequenza. La FFT ci fornisce una rappresentazione a valori complessi che include sia informazioni di ampiezza che di fase. Tuttavia, per gli MFCC, solo l'ampiezza viene utilizzata per calcolare lo spettro di potenza. Lo spettro di potenza è il quadrato dello spettro di ampiezza e misura l'energia presente in ogni componente di frequenza.

Lo spettro di potenza $P(f)$ di un segnale $x(t)$ è definito come $P(f) = |X(f)|^2$, dove $X(f)$ è la trasformata di Fourier di $x(t)$. Elevando al quadrato l'ampiezza della trasformata di Fourier, mettiamo in risalto le frequenze *più forti* rispetto a quelle *più deboli*, catturando così caratteristiche spettrali più rilevanti del segnale audio. Ciò è importante in applicazioni come la classificazione audio, il riconoscimento vocale e il Keyword Spotting (KWS), in cui l'attenzione è rivolta all'identificazione di distinti pattern di frequenza che caratterizzano diverse classi di audio o fonemi nel parlato.

- **Mel Filter Banks**: Il dominio di frequenza viene poi mappato sulla scala Mel, che approssima la risposta dell'orecchio umano a diverse frequenze. L'idea è di estrarre più feature (più filtri) nelle frequenze più basse e meno in quelle alte. Pertanto, funziona bene sui suoni distinti dall'orecchio umano. In genere, da 20 a 40 filtri triangolari estraggono le energie di frequenza Mel. Queste energie vengono poi trasformate in logaritmo per convertire i fattori moltiplicativi in quelli additivi, rendendoli più adatti per un'ulteriore elaborazione.

- **Discrete Cosine Transform (DCT)**: L'ultimo passaggio consiste nell'applicare la Trasformata discreta del coseno (DCT) alle energie del log Mel. La DCT aiuta a de-correlare le energie, comprimendo efficacemente i dati e mantenendo solo le feature più discriminanti. Di solito, vengono mantenuti i primi 12-13 coefficienti DCT, formando il vettore finale delle feature MFCC.

Pratica con Python

Applichiamo quanto discusso lavorando su un campione audio reale. Aprire il notebook su Google CoLab ed estrarre le funzionalità MLCC sui campioni audio: [Open In Colab]

Conclusione

Quale Tecnica di Estrazione delle Feature dovremmo usare?

I Mel-frequency Cepstral Coefficient (MFCC), le Mel-Frequency Energie (MFE), o lo Spettrogramma sono tecniche per rappresentare dati audio, che sono spesso utili in diversi contesti.

In generale, gli MFCC sono più focalizzati sulla cattura dell'inviluppo dello spettro di potenza, il che li rende meno sensibili ai dettagli spettrali a grana fine ma più robusti al rumore. Questo è spesso auspicabile per le attività relative al parlato. D'altro canto, gli spettrogrammi o MFE conservano informazioni di frequenza più dettagliate, il che può essere vantaggioso in attività che richiedono discriminazione basata su contenuti spettrali a grana fine.

Gli MFCC sono particolarmente efficaci per:

1. **Riconoscimento Vocale**: Gli MFCC sono eccellenti per identificare il contenuto fonetico nei segnali vocali.
2. **Identificazione del Parlante**: Possono essere utilizzati per distinguere tra diversi parlanti in base alle caratteristiche della voce.
3. **Riconoscimento delle Emozioni**: Gli MFCC possono catturare le sfumature nel parlato indicative di stati emotivi.
4. **Keyword Spotting**: Specialmente in TinyML, dove la bassa complessità computazionale e le piccole dimensioni delle funzionalità sono cruciali.

Gli spettrogrammi o MFE sono spesso più adatti per:

1. **Analisi Musicale**: Gli spettrogrammi possono catturare strutture armoniche e timbriche nella musica, il che è essenziale per attività come la classificazione del genere, il riconoscimento degli strumenti o la trascrizione musicale.
2. **Classificazione dei Suoni Ambientali**: Nel riconoscimento di suoni ambientali non vocali (ad esempio pioggia, vento, traffico), lo spettrogramma completo può fornire feature più discriminanti.
3. **Identificazione del Canto degli Uccelli**: I dettagli intricati dei richiami degli uccelli sono spesso meglio catturati utilizzando gli spettrogrammi.

4. **Elaborazione del Segnale Bioacustico**: In applicazioni come l'analisi dei richiami dei delfini o dei pipistrelli, le informazioni di frequenza a grana fine in uno spettrogramma possono essere essenziali.
5. **Audio Quality Assurance**: Gli spettrogrammi sono spesso utilizzati nell'analisi audio professionale per identificare rumori indesiderati, clic o altri artefatti.

Risorse

- Notebook Audio_Data_Analysis Colab

Blocco delle Feature Spettrali DSP

DALL·E 3 Prompt: Illustrazione in stile fumetto anni '50 di uno scienziato latino e una scienziata in una sala di ricerca sulle vibrazioni. L'uomo sta usando un righello per esaminare vecchi circuiti. La donna è al computer con grafici complessi sulle vibrazioni. Il tavolo di legno ha delle schede con dei sensori, in primo piano un accelerometro. Un classico computer con la parte posteriore arrotondata mostra l'IDE Arduino con codice per le assegnazioni dei pin LED e algoritmi di apprendimento automatico per il rilevamento del movimento. Il Serial Monitor visualizza FFT, classificazione, wavelet e DSP. Lampade vintage, strumenti e grafici FFT e Wavelet completano la scena.

Introduzione

I progetti TinyML correlati al movimento (o alle vibrazioni) coinvolgono dati da IMU (solitamente **accelerometri** e **giroscopi**). Questi dataset di tipo temporale dovrebbero essere pre-elaborati prima di inserirli in un training di modello di apprendimento automatico, che è un'area impegnativa per l'apprendimento automatico embedded. Tuttavia, Edge Impulse aiuta a superare questa complessità con la sua fase di pre-elaborazione dell'elaborazione del segnale digitale (DSP) e, più specificamente, il Spectral Features Block per i sensori inerziali.

Ma come funziona internamente? Analizziamolo nel dettaglio.

Estrazione delle Feature di Revisione

L'estrazione di feature caratteristiche da un dataset catturato con sensori inerziali, come gli accelerometri, comporta l'elaborazione e l'analisi dei dati grezzi. Gli accelerometri misurano l'accelerazione di un oggetto lungo uno o più assi (in genere tre, indicati come X, Y e Z). Queste misure possono essere utilizzate per comprendere vari aspetti del movimento dell'oggetto, come pattern di movimento e vibrazioni. Ecco una panoramica di alto livello del processo:

Raccolta dati: Per prima cosa, dobbiamo raccogliere i dati dagli accelerometri. A seconda dell'applicazione, i dati possono essere raccolti a diverse frequenze di campionamento. È essenziale assicurarsi che la frequenza di campionamento sia sufficientemente alta da catturare le dinamiche rilevanti del movimento studiato (la frequenza di campionamento dovrebbe essere almeno il doppio della massima frequenza presente nel segnale).

Pre-elaborazione dei dati: I dati grezzi dell'accelerometro possono essere rumorosi e contenere errori o informazioni irrilevanti. I passaggi di pre-elaborazione, come il filtraggio e la normalizzazione, possono aiutare a pulire e standardizzare i dati, rendendoli più adatti all'estrazione di feature.

> Studio non esegue la normalizzazione o la standardizzazione, quindi a volte, quando si lavora con Sensor Fusion [gruppi di sensori], potrebbe essere necessario eseguire questo passaggio prima di caricare i dati in Studio. Ciò è particolarmente cruciale nei progetti di gruppi di sensori, come visto in questo tutorial, Sensor Data Fusion with Spresense and CommonSense.

Segmentazione: A seconda della natura dei dati e dell'applicazione, potrebbe essere necessario dividere i dati in segmenti più piccoli o **finestre**. Ciò può aiutare a concentrarsi su eventi o attività specifici all'interno del dataset, rendendo l'estrazione di feature più gestibile e significativa. La scelta della **window size** [dimensione della finestra] e della sovrapposizione (**window span**) dipende dall'applicazione e dalla frequenza degli eventi di interesse. Come regola generale, dovremmo provare a catturare un paio di "cicli di dati".

Estrazione delle feature: Una volta che i dati sono stati pre-elaborati e segmentati, è possibile estrarre feature che descrivono le caratteristiche del movimento. Alcune feature tipiche estratte dai dati dell'accelerometro includono:

- Le feature del **Time-domain** descrivono le proprietà statistiche dei dati all'interno di ciascun segmento, come media, mediana, deviazione standard, asimmetria, curtosi e tasso di attraversamento dello zero.
- Le feature **Frequency-domain** si ottengono trasformando i dati nel dominio della frequenza utilizzando tecniche come la Fast Fourier Transform (FFT). Alcune feature tipiche del dominio della frequenza includono lo spettro di potenza, l'energia spettrale, le frequenze dominanti (ampiezza e frequenza) e l'entropia spettrale.
- Le feature del dominio **Time-frequency** combinano le informazioni del dominio del tempo e della frequenza, come la Short-Time Fourier Transform (STFT) o la Discrete Wavelet Transform (DWT). Possono fornire una comprensione più dettagliata di come il contenuto di frequenza del segnale cambia nel tempo.

In molti casi, il numero di feature estratte può essere elevato, il che può portare a un overfitting o a una maggiore complessità computazionale. Le tecniche di selezione delle feature, come le informazioni reciproche, i metodi basati sulla correlazione o l'analisi delle componenti principali (PCA), possono aiutare a identificare le feature più rilevanti per una determinata applicazione e ridurre la dimensionalità del dataset. Studio può aiutare con tali calcoli rilevanti per le feature.

Esploriamo più in dettaglio un tipico progetto di classificazione del movimento TinyML trattato in questa serie di esercitazioni pratiche.

Un progetto TinyML di Motion Classification

Nel progetto, *Motion Classification and Anomaly Detection*, abbiamo simulato sollecitazioni meccaniche nel trasporto, dove il nostro problema era classificare quattro classi di movimento:

- **Maritime** (Pallet in navi)
- **Terrestrial** (pallet in un Camion o Treno)
- **Lift** [Sollevamento] (pallet movimentati da Carrello elevatore)
- **Idle** (pallet in Magazzini)

Gli accelerometri hanno fornito i dati sul pallet (o il container).

Di seguito è riportato un campione (dati grezzi) di 10 secondi, acquisito con una frequenza di campionamento di 50 Hz:

Il risultato è simile quando questa analisi viene eseguita su un altro set di dati con lo stesso principio, utilizzando una frequenza di campionamento diversa, 62.5Hz invece di 50Hz.

Pre-elaborazione dei Dati

I dati grezzi acquisiti dall'accelerometro (dati di "serie temporali") devono essere convertiti in "dati tabellari" utilizzando uno dei tipici metodi di estrazione delle feature descritti nell'ultima sezione.
Dovremmo segmentare i dati utilizzando una finestra scorrevole sui dati campione per l'estrazione delle feature. Il progetto ha acquisito i dati dell'accelerometro ogni 10 secondi con una frequenza di campionamento di 62.5 Hz. Una finestra di 2 secondi acquisisce 375 punti dati (3 assi x 2 secondi x 62.5 campioni). La finestra scorre ogni 80 ms, creando un dataset più grande in cui ogni istanza ha 375 "feature grezze".

Su Studio, la versione precedente (V1) dello **Spectral Analysis Block** estraeva come caratteristiche del dominio del tempo solo l'RMS e, per il dominio della frequenza, i picchi e la frequenza (utilizzando FFT) e le caratteristiche di potenza (PSD) del segnale nel tempo, risultando in un set di dati tabellari fisso di 33 caratteristiche (11 per ogni asse),

Quelle 33 feature erano il tensore di input di un classificatore di reti neurali.
Nel 2022, Edge Impulse ha rilasciato la versione 2 dello Spectral Analysis block, che esploreremo qui.

Edge Impulse - Spectral Analysis Block V.2 funzionamento

Nella Versione 2, le caratteristiche statistiche del dominio del tempo per asse/canale sono:

- RMS
- Skewness
- Kurtosis

E le caratteristiche spettrali del dominio della frequenza per asse/canale sono:

- Spectral Power [Potenza spettrale]
- Skewness [asimmetria] (nella prossima versione)
- Curtosi (nella prossima versione)

In questo link possiamo avere maggiori dettagli sull'estrazione delle feature.

> Clonare il progetto pubblico. Si può anche seguire la spiegazione, giocando col codice usando il mio Google CoLab Notebook: Edge Impulse Spectral Analysis Block Notebook.

Si inizia importando le librerie:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import math
from scipy.stats import skew, kurtosis
from scipy import signal
from scipy.signal import welch
from scipy.stats import entropy
from sklearn import preprocessing
import pywt

plt.rcParams['figure.figsize'] = (12, 6)
plt.rcParams['lines.linewidth'] = 3
```

Dal progetto studiato, scegliamo un campione di dati dagli accelerometri come di seguito:

- Dimensione della finestra di 2 secondi: `[2,000]` ms
- Frequenza di campionamento: `[62.5]` Hz
- Sceglieremo il filtro `[None]` (per semplicità) e una
- Lunghezza FFT: `[16]`.

```
f =  62.5 # Hertz
wind_sec = 2 # seconds
FFT_Lenght = 16
axis = ['accX', 'accY', 'accZ']
n_sensors = len(axis)
```

Selezionando *Raw Features* nella scheda Studio Spectral Analysis, possiamo copiare tutti i 375 punti dati di una particolare finestra di 2 secondi negli appunti.

Incollare i punti dati in una nuova variabile *data*:

```
data=[-5.6330, 0.2376, 9.8701, -5.9442, 0.4830, 9.8701, -5.4217, ...]
No_raw_features = len(data)
N = int(No_raw_features/n_sensors)
```

Le feature grezze totali sono 375, ma lavoreremo con ogni asse singolarmente, dove N= 125 (numero di campioni per asse).
Vogliamo capire come Edge Impulse ottiene le feature elaborate.

Quindi, si devono anche incollare le feature elaborate su una variabile (per confrontare le feature calcolate in Python con quelle fornite da Studio):

```
features = [2.7322, -0.0978, -0.3813, 2.3980, 3.8924, 24.6841, 9.6303, ...]
N_feat = len(features)
N_feat_axis = int(N_feat/n_sensors)
```

Il numero totale di feature elaborate è 39, il che significa 13 feature/asse.
Osservando attentamente queste 13 feature, ne troveremo 3 per il dominio del tempo (RMS, Skewness e Kurtosis):

- `[rms] [skew] [kurtosis]`

e 10 per il dominio della frequenza (ci torneremo più avanti).

- `[spectral skew][spectral kurtosis][Spectral Power 1] ... [Spectral Power 8]`

Suddivisione dei dati grezzi per sensore

I dati hanno campioni da tutti gli assi; dividiamoli e tracciamo un grafico separatamente:

```
def plot_data(sensors, axis, title):
    [plt.plot(x, label=y) for x,y in zip(sensors, axis)]
    plt.legend(loc='lower right')
    plt.title(title)
    plt.xlabel('#Sample')
    plt.ylabel('Value')
    plt.box(False)
    plt.grid()
    plt.show()

accX = data[0::3]
accY = data[1::3]
accZ = data[2::3]
sensors = [accX, accY, accZ]
plot_data(sensors, axis, 'Raw Features')
```

Sottrazione della media

Successivamente, dovremmo sottrarre la media dai *dati*. La sottrazione della media da un set di dati è una comune fase di pre-elaborazione dei dati in statistica e apprendimento

automatico. Lo scopo della sottrazione della media dai dati è di centrare i dati attorno allo zero. Questo è importante perché può rivelare pattern e relazioni che potrebbero essere nascosti se i dati non sono centrati.
Ecco alcuni motivi specifici per cui la sottrazione della media può essere utile:

- Semplifica l'analisi: Centrando i dati, la media diventa zero, rendendo alcuni calcoli più semplici e facili da interpretare.
- Rimuove la distorsione: Se i dati sono distorti, sottraendo la media è possibile rimuoverli e consentire un'analisi più accurata.
- Può rivelare pattern: Centrare i dati può aiutare a scoprire pattern che potrebbero essere nascosti se i dati non sono centrati. Ad esempio, centrare i dati può aiutare a identificare i trend nel tempo se si analizza un set di dati di serie temporali.
- Può migliorare le prestazioni: in alcuni algoritmi di apprendimento automatico, centrare i dati può migliorare le prestazioni riducendo l'influenza dei valori anomali e rendendo i dati più facilmente confrontabili. Nel complesso, sottrarre la media è una tecnica semplice ma potente che può essere utilizzata per migliorare l'analisi e l'interpretazione dei dati.

```
dtmean = [(sum(x)/len(x)) for x in sensors]
[print('mean_'+x+'= ', round(y, 4)) for x,y in zip(axis, dtmean)][0]

accX = [(x - dtmean[0]) for x in accX]
accY = [(x - dtmean[1]) for x in accY]
accZ = [(x - dtmean[2]) for x in accZ]
sensors = [accX, accY, accZ]

plot_data(sensors, axis, 'Raw Features - Subctract the Mean')
```

Feature Statistiche del Dominio del Tempo

Calcolo RMS

Il valore RMS di un set di valori (o di una forma d'onda a tempo continuo) è la radice quadrata della media aritmetica dei quadrati dei valori o del quadrato della funzione che definisce la forma d'onda continua. In fisica, il valore RMS di una corrente elettrica è definito come il "valore della corrente continua che dissipa la stessa potenza in un resistore".
Nel caso di un set di n valori $\{x1, x2, \ldots, xn\}$, l'RMS è:

$$x_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)}\ .$$

NOTARE che il valore RMS è diverso per i dati grezzi originali e dopo aver sottratto la media

```
# Using numpy and standartized data (subtracting mean)
rms = [np.sqrt(np.mean(np.square(x))) for x in sensors]
```

Possiamo confrontare i valori RMS calcolati qui con quelli presentati da Edge Impulse:

```
[print('rms_'+x+'= ', round(y, 4)) for x,y in zip(axis, rms)][0]
print("\nCompare with Edge Impulse result features")
print(features[0:N_feat:N_feat_axis])
```

```
rms_accX=  2.7322
rms_accY=  0.7833
rms_accZ=  0.1383
```

Rispetto alle caratteristiche dei risultati di Edge Impulse:

```
[2.7322, 0.7833, 0.1383]
```

Calcolo di asimmetria e curtosi

In statistica, asimmetria e curtosi sono due modi per misurare la **forma di una distribuzione**. Qui possiamo vedere la distribuzione dei valori del sensore:

```
fig, axes = plt.subplots(nrows=1, ncols=3, figsize=(13, 4))
sns.kdeplot(accX, fill=True, ax=axes[0])
sns.kdeplot(accY, fill=True, ax=axes[1])
sns.kdeplot(accZ, fill=True, ax=axes[2])
axes[0].set_title('accX')
axes[1].set_title('accY')
axes[2].set_title('accZ')
plt.suptitle('IMU Sensors distribution', fontsize=16, y=1.02)
plt.show()
```

La **Skewness** [asimmetria] è una misura dell'asimmetria di una distribuzione. Questo valore può essere positivo o negativo.

- Un'asimmetria negativa indica che la coda si trova sul lato sinistro della distribuzione, che si estende verso valori più negativi.
- Un'asimmetria positiva indica che la coda si trova sul lato destro della distribuzione, che si estende verso valori più positivi.
- Un valore zero indica che non c'è alcuna asimmetria nella distribuzione, il che significa che la distribuzione è perfettamente simmetrica.

```
skew = [skew(x, bias=False) for x in sensors]
[print('skew_'+x+'= ', round(y, 4)) for x,y in zip(axis, skew)][0]
print("\nCompare with Edge Impulse result features")
features[1:N_feat:N_feat_axis]
```

```
skew_accX=  -0.099
skew_accY=  0.1756
skew_accZ=  6.9463
```

Rispetto alle caratteristiche dei risultati di Edge Impulse:

```
[-0.0978, 0.1735, 6.8629]
```

La **Kurtosis** è una misura che indica se una distribuzione è a coda pesante o a coda leggera rispetto a una distribuzione normale.

- La curtosi di una distribuzione normale è zero.
- Se una data distribuzione ha una curtosi negativa, si dice che è platicurtica, il che significa che tende a produrre meno valori anomali e meno estremi rispetto alla distribuzione normale.
- Se una data distribuzione ha una curtosi positiva, si dice che è leptocurtica, il che significa che tende a produrre più valori anomali rispetto alla distribuzione normale.

```
kurt = [kurtosis(x, bias=False) for x in sensors]
[print('kurt_'+x+'= ', round(y, 4)) for x,y in zip(axis, kurt)][0]
print("\nCompare with Edge Impulse result features")
features[2:N_feat:N_feat_axis]
```

```
kurt_accX=  -0.3475
kurt_accY=  1.2673
kurt_accZ=  68.1123
```

Rispetto alle caratteristiche dei risultati di Edge Impulse:

```
[-0.3813, 1.1696, 65.3726]
```

Feature spettrali

Il segnale filtrato viene trasmesso alla sezione di potenza spettrale, che calcola la **FFT** per generare le caratteristiche spettrali.
Poiché la finestra campionata è solitamente più grande della dimensione FFT, la finestra verrà suddivisa in frame (o "sotto-finestre") e la FFT verrà calcolata su ogni frame.

FFT length - La dimensione della FFT. Questo determina il numero di bin FFT e la risoluzione dei picchi di frequenza che possono essere separati. Un numero basso significa che più segnali saranno mediati insieme nello stesso bin FFT, ma riduce anche il numero di feature e la dimensione del modello. Un numero alto separerà più segnali in bin separati, generando un modello più grande.

- Il numero totale di feature di potenza spettrale varierà a seconda di come si impostano i parametri del filtro e di FFT. Senza filtro, il numero di feature è 1/2 della lunghezza FFT.

Spectral Power - Metodo di Welch

Dovremmo usare il metodo di Welch per dividere il segnale nel dominio della frequenza in "bin" e calcolare lo spettro di potenza per ogni bin. Questo metodo divide il segnale in segmenti sovrapposti, applica una funzione finestra a ogni segmento, calcola il periodogramma di ogni segmento usando DFT e ne fa la media per ottenere una stima più uniforme dello spettro di potenza.

```
# Function used by Edge Impulse instead of scipy.signal.welch().
def welch_max_hold(fx, sampling_freq, nfft, n_overlap):
    n_overlap = int(n_overlap)
    spec_powers = [0 for _ in range(nfft//2+1)]
    ix = 0
    while ix <= len(fx):
        # Slicing truncates if end_idx > len, and rfft will auto-zero pad
        fft_out = np.abs(np.fft.rfft(fx[ix:ix+nfft], nfft))
        spec_powers = np.maximum(spec_powers, fft_out**2/nfft)
        ix = ix + (nfft-n_overlap)
    return np.fft.rfftfreq(nfft, 1/sampling_freq), spec_powers
```

Applicazione della funzione di cui sopra a 3 segnali:

```
fax,Pax = welch_max_hold(accX, fs, FFT_Lenght, 0)
fay,Pay = welch_max_hold(accY, fs, FFT_Lenght, 0)
faz,Paz = welch_max_hold(accZ, fs, FFT_Lenght, 0)
specs = [Pax, Pay, Paz ]
```

Possiamo tracciare lo spettro di potenza P(f):

```
plt.plot(fax,Pax, label='accX')
plt.plot(fay,Pay, label='accY')
plt.plot(faz,Paz, label='accZ')
plt.legend(loc='upper right')
plt.xlabel('Frequency (Hz)')
#plt.ylabel('PSD [V**2/Hz]')
plt.ylabel('Power')
plt.title('Power spectrum P(f) using Welch's method')
plt.grid()
plt.box(False)
plt.show()
```

Oltre allo spettro di potenza, possiamo anche includere l'asimmetria e la curtosi delle feature nel dominio della frequenza (dovrebbero essere disponibili in una nuova versione):

```
spec_skew = [skew(x, bias=False) for x in specs]
spec_kurtosis = [kurtosis(x, bias=False) for x in specs]
```

Elenchiamo ora tutte le feature spettrali per asse e confrontarli con EI:

```
print("EI Processed Spectral features (accX): ")
print(features[3:N_feat_axis][0:])
print("\nCalculated features:")
print (round(spec_skew[0],4))
print (round(spec_kurtosis[0],4))
[print(round(x, 4)) for x in Pax[1:]][0]
```

EI Processed Spectral features (accX):
2.398, 3.8924, 24.6841, 9.6303, 8.4867, 7.7793, 2.9963, 5.6242, 3.4198, 4.2735
Calculated features:
2.9069 8.5569 24.6844 9.6304 8.4865 7.7794 2.9964 5.6242 3.4198 4.2736

```
print("EI Processed Spectral features (accY): ")
print(features[16:26][0:]) #13: 3+N_feat_axis;  26 = 2x N_feat_axis
print("\nCalculated features:")
print (round(spec_skew[1],4))
print (round(spec_kurtosis[1],4))
[print(round(x, 4)) for x in Pay[1:]][0]
```

EI Processed Spectral features (accY):
0.9426, -0.8039, 5.429, 0.999, 1.0315, 0.9459, 1.8117, 0.9088, 1.3302, 3.112
Calculated features:
1.1426 -0.3886 5.4289 0.999 1.0315 0.9458 1.8116 0.9088 1.3301 3.1121

```
print("EI Processed Spectral features (accZ): ")
print(features[29:][0:]) #29: 3+(2*N_feat_axis);
print("\nCalculated features:")
print (round(spec_skew[2],4))
print (round(spec_kurtosis[2],4))
[print(round(x, 4)) for x in Paz[1:]][0]
```

EI Processed Spectral features (accZ):
0.3117, -1.3812, 0.0606, 0.057, 0.0567, 0.0976, 0.194, 0.2574, 0.2083, 0.166
Calculated features:
0.3781 -1.4874 0.0606 0.057 0.0567 0.0976 0.194 0.2574 0.2083 0.166

Dominio tempo-frequenza

Wavelet

Wavelet è una tecnica potente per analizzare segnali con feature transitorie o bruschi cambiamenti, come picchi o bordi, che sono difficili da interpretare con i metodi tradizionali basati su Fourier.

Le trasformazioni wavelet funzionano scomponendo un segnale in diverse componenti di frequenza e analizzandole individualmente. La trasformazione si ottiene convolvendo il segnale con una **funzione wavelet**, una piccola forma d'onda centrata su un tempo e una frequenza specifici. Questo processo scompone efficacemente il segnale in diverse bande di frequenza, ciascuna delle quali può essere analizzata separatamente.
Uno dei vantaggi fondamentali delle trasformazioni wavelet è che consentono l'analisi tempo-frequenza, il che significa che possono rivelare il contenuto di frequenza di un segnale mentre cambia nel tempo. Ciò le rende particolarmente utili per analizzare segnali non stazionari, che variano nel tempo.
Le wavelet hanno molte applicazioni pratiche, tra cui la compressione di segnali e immagini, il denoising [rimozione del rumore], l'estrazione di feature e l'elaborazione di immagini.
Selezioniamo Wavelet sul blocco Spectral Features nello stesso progetto:

- Type: Wavelet

- Wavelet Decomposition Level: 1
- Wavelet: bior1.3

La Funzione Wavelet

```
wavelet_name='bior1.3'
num_layer = 1

wavelet = pywt.Wavelet(wavelet_name)
[phi_d,psi_d,phi_r,psi_r,x] = wavelet.wavefun(level=5)
plt.plot(x, psi_d, color='red')
plt.title('Wavelet Function')
plt.ylabel('Value')
plt.xlabel('Time')
plt.grid()
plt.box(False)
plt.show()
```

Come abbiamo fatto prima, copiamo e incolliamo le Processed Features:

```
features = [3.6251, 0.0615, 0.0615, -7.3517, -2.7641, 2.8462, 5.0924, ...]
N_feat = len(features)
N_feat_axis = int(N_feat/n_sensors)
```

Edge Impulse calcola la Discrete Wavelet Transform (DWT) per ciascuno dei livelli di Wavelet Decomposition selezionati. Dopodiché, le feature verranno estratte.

Nel caso di **Wavelet**, le feature estratte sono *valori statistici di base*, *valori di incrocio* ed *entropia*. Ci sono, in totale, 14 feature per layer come di seguito:

- [11] Feature Statistiche: **n5, n25, n75, n95, media, mediana,** deviazione standard **(std)**, varianza **(var)** radice quadrata media **(rms), curtosi** e asimmetria **(skew)**.
- [2] Feature di attraversamento: Il tasso di attraversamento dello zero **(zcross)** e il tasso di attraversamento medio **(mcross)** sono rispettivamente i tempi in cui il segnale attraversa la linea di base (y = 0) e il livello medio (y = u) per unità di tempo
- [1] Feature di Complessità: L'**entropia** è una misura caratteristica della complessità del segnale

Tutti i 14 valori sopra indicati vengono calcolati per ogni Layer (incluso L0, il segnale originale)

- Il numero totale di feature varia a seconda di come si imposta il filtro e del numero di layer. Ad esempio, con il filtro [None] e Level[1], il numero di feature per asse sarà 14 x 2 (L0 e L1) = 28. Per i tre assi, avremo un totale di 84 feature.

Analisi Wavelet

L'analisi wavelet scompone il segnale (**accX, accY**, **e accZ**) in diverse componenti di frequenza utilizzando un set di filtri, che separano queste componenti in componenti a bassa frequenza (parti del segnale che variano lentamente e contengono pattern a lungo termine), come **accX_l1, accY_l1, accZ_l1** e componenti ad alta frequenza (parti del segnale che

variano rapidamente e contengono modelli a breve termine), come **accX_d1, accY_d1, accZ_d1**, consentendo l'estrazione di feature per ulteriori analisi o classificazioni. Verranno utilizzati solo le componenti a bassa frequenza (coefficienti di approssimazione o cA). In questo esempio, assumiamo un solo livello (Single-level Discrete Wavelet Transform), in cui la funzione restituirà una tupla. Con una decomposizione multilivello, la "Multilevel 1D Discrete Wavelet Transform", il risultato sarà un elenco (per i dettagli, vedere: Discrete Wavelet Transform (DWT))

```
(accX_l1, accX_d1) = pywt.dwt(accX, wavelet_name)
(accY_l1, accY_d1) = pywt.dwt(accY, wavelet_name)
(accZ_l1, accZ_d1) = pywt.dwt(accZ, wavelet_name)
sensors_l1 = [accX_l1, accY_l1, accZ_l1]

# Plot power spectrum versus frequency
plt.plot(accX_l1, label='accX')
plt.plot(accY_l1, label='accY')
plt.plot(accZ_l1, label='accZ')
plt.legend(loc='lower right')
plt.xlabel('Time')
plt.ylabel('Value')
plt.title('Wavelet Approximation')
plt.grid()
plt.box(False)
plt.show()
```

Estrazione delle Feature

Cominciamo con le feature statistiche di base. Notare che applichiamo la funzione sia per i segnali originali che per i cAs risultanti dal DWT:

```
def calculate_statistics(signal):
    n5 = np.percentile(signal, 5)
    n25 = np.percentile(signal, 25)
    n75 = np.percentile(signal, 75)
    n95 = np.percentile(signal, 95)
    median = np.percentile(signal, 50)
    mean = np.mean(signal)
    std = np.std(signal)
    var = np.var(signal)
    rms = np.sqrt(np.mean(np.square(signal)))
    return [n5, n25, n75, n95, median, mean, std, var, rms]

stat_feat_l0 = [calculate_statistics(x) for x in sensors]
stat_feat_l1 = [calculate_statistics(x) for x in sensors_l1]
```

Asimmetria e Curtosi:

```
skew_l0 = [skew(x, bias=False) for x in sensors]
skew_l1 = [skew(x, bias=False) for x in sensors_l1]
kurtosis_l0 = [kurtosis(x, bias=False) for x in sensors]
kurtosis_l1 = [kurtosis(x, bias=False) for x in sensors_l1]
```

Zero crossing (zcross) è il numero di volte in cui il coefficiente wavelet attraversa l'asse dello zero. Può essere utilizzato per misurare il contenuto di frequenza del segnale poiché i segnali ad alta frequenza tendono ad avere più attraversamenti per lo zero rispetto ai segnali a bassa frequenza.

Mean crossing (mcross), d'altra parte, è il numero di volte in cui il coefficiente wavelet attraversa la media del segnale. Può essere utilizzato per misurare l'ampiezza poiché i segnali ad alta ampiezza tendono ad avere più attraversamenti medi rispetto ai segnali a bassa ampiezza.

```
def getZeroCrossingRate(arr):
    my_array = np.array(arr)
    zcross = float("{0:.2f}".format((((my_array[:-1] * my_array[1:]) < 0).su    m())/len(
arr)))
    return zcross

def getMeanCrossingRate(arr):
    mcross = getZeroCrossingRate(np.array(arr) - np.mean(arr))
    return mcross

def calculate_crossings(list):
    zcross=[]
    mcross=[]
    for i in range(len(list)):
        zcross_i = getZeroCrossingRate(list[i])
        zcross.append(zcross_i)
        mcross_i = getMeanCrossingRate(list[i])
        mcross.append(mcross_i)
    return zcross, mcross

cross_l0 = calculate_crossings(sensors)
cross_l1 = calculate_crossings(sensors_l1)
```

Nell'analisi wavelet, l'**entropia** si riferisce al grado di disordine o casualità nella distribuzione dei coefficienti wavelet. Qui, abbiamo utilizzato l'entropia di Shannon, che misura l'incertezza o la casualità di un segnale. Viene calcolata come la somma negativa delle probabilità dei diversi possibili risultati del segnale moltiplicata per il loro logaritmo in base 2. Nel contesto dell'analisi wavelet, l'entropia di Shannon può essere utilizzata per misurare la complessità del segnale, con valori più alti che indicano una maggiore complessità.

```
def calculate_entropy(signal, base=None):
    value, counts = np.unique(signal, return_counts=True)
    return entropy(counts, base=base)

entropy_l0 = [calculate_entropy(x) for x in sensors]
entropy_l1 = [calculate_entropy(x) for x in sensors_l1]
```

Elenchiamo ora tutte le feature wavelet e creiamo un elenco per layer.

```
L1_features_names = ["L1-n5", "L1-n25", "L1-n75", "L1-n95", "L1-median", "L1-mean", "L1-s
td", "L1-var", "L1-rms", "L1-skew", "L1-Kurtosis", "L1-zcross", "L1-mcross", "L1-entropy"
]

L0_features_names = ["L0-n5", "L0-n25", "L0-n75", "L0-n95", "L0-median", "L0-mean", "L0-s
td", "L0-var", "L0-rms", "L0-skew", "L0-Kurtosis", "L0-zcross", "L0-mcross", "L0-entropy"
]

all_feat_l0 = []
for i in range(len(axis)):
    feat_l0 = stat_feat_l0[i]+[skew_l0[i]]+[kurtosis_l0[i]]+[cross_l0[0][i]]+[cross_l0[1]
[i]]+[entropy_l0[i]]
    [print(axis[i]+' '+x+'= ', round(y, 4)) for x,y in zip(L0_features_names, feat_l0)][0
]
    all_feat_l0.append(feat_l0)
all_feat_l0 = [item for sublist in all_feat_l0 for item in sublist]
```

```
print(f"\nAll L0 Features = {len(all_feat_l0)}")

all_feat_l1 = []
for i in range(len(axis)):
feat_l1 = stat_feat_l1[i]+[skew_l1[i]]+[kurtosis_l1[i]]+[cross_l1[0][i]]+[cross_l1[1][i]]
+[entropy_l1[i]]
[print(axis[i]+' '+x+'= ', round(y, 4)) for x,y in zip(L1_features_names, feat_l1)][0]
all_feat_l1.append(feat_l1)
all_feat_l1 = [item for sublist in all_feat_l1 for item in sublist]
print(f"\nAll L1 Features = {len(all_feat_l1)}")
```

```
accX L0-n5=  -4.9364
accX L0-n25=  -1.8429
accX L0-n75=  1.8842
accX L0-n95=  3.8096
accX L0-median=  0.4058
accX L0-mean=  -0.0
accX L0-std=  2.7322
accX L0-var=  7.4651
accX L0-rms=  2.7322
accX L0-skew=  -0.099
accX L0-Kurtosis=  -0.3475
accX L0-zcross=  0.06
accX L0-mcross=  0.06
accX L0-entropy=  4.8283
accY L0-n5=  -1.149
accY L0-n25=  -0.4475
accY L0-n75=  0.4814
accY L0-n95=  1.1491
accY L0-median=  -0.0315
accY L0-mean=  0.0
accY L0-std=  0.7833
accY L0-var=  0.6136
accY L0-rms=  0.7833
accY L0-skew=  0.1756
accY L0-Kurtosis=  1.2673
accY L0-zcross=  0.29
accY L0-mcross=  0.29
accY L0-entropy=  4.8283
accZ L0-n5=  -0.1242
accZ L0-n25=  -0.0429
accZ L0-n75=  0.0349
accZ L0-n95=  0.0839
accZ L0-median=  -0.0112
accZ L0-mean=  0.0
accZ L0-std=  0.1383
accZ L0-var=  0.0191
accZ L0-rms=  0.1383
accZ L0-skew=  6.9463
accZ L0-Kurtosis=  68.1123
accZ L0-zcross=  0.35
accZ L0-mcross=  0.35
accZ L0-entropy=  4.5649

All L0 Features = 42
```

```
accX L1-n5=  -7.3516
accX L1-n25=  -2.7641
accX L1-n75=  2.8462
accX L1-n95=  5.0924
accX L1-median=  0.4064
accX L1-mean=  -0.2133
accX L1-std=  3.8473
accX L1-var=  14.8015
accX L1-rms=  3.8532
accX L1-skew=  -0.2975
accX L1-Kurtosis=  -0.7631
accX L1-zcross=  0.06
accX L1-mcross=  0.06
accX L1-entropy=  4.1744
accY L1-n5=  -1.3234
accY L1-n25=  -0.6492
accY L1-n75=  0.7844
accY L1-n95=  1.361
accY L1-median=  0.0659
accY L1-mean=  0.0276
accY L1-std=  0.9345
accY L1-var=  0.8732
accY L1-rms=  0.9349
accY L1-skew=  0.2874
accY L1-Kurtosis=  0.0347
accY L1-zcross=  0.31
accY L1-mcross=  0.31
accY L1-entropy=  4.1317
accZ L1-n5=  -0.1126
accZ L1-n25=  -0.0493
accZ L1-n75=  0.0348
accZ L1-n95=  0.1022
accZ L1-median=  -0.0137
accZ L1-mean=  0.0025
accZ L1-std=  0.1053
accZ L1-var=  0.0111
accZ L1-rms=  0.1053
accZ L1-skew=  4.4095
accZ L1-Kurtosis=  28.6586
accZ L1-zcross=  0.4
accZ L1-mcross=  0.37
accZ L1-entropy=  4.1531

All L1 Features = 42
```

Conclusione

Edge Impulse Studio è una potente piattaforma online in grado di gestire per noi l'attività di pre-elaborazione. Tuttavia, data la nostra prospettiva ingegneristica, vogliamo capire cosa sta succedendo "sotto il cofano". Questa conoscenza ci aiuterà a trovare le migliori opzioni e gli iperparametri per ottimizzare i nostri progetti.

Daniel Situnayake ha scritto nel suo blog: "I dati grezzi dei sensori sono altamente dimensionali e rumorosi. Gli algoritmi di elaborazione del segnale digitale ci aiutano a separare il segnale dal rumore. Il DSP è una parte essenziale dell'ingegneria embedded e molti processori edge hanno un'accelerazione integrata per il DSP. Come ingegnere ML, imparare il DSP di base dà dei superpoteri per gestire dati di serie temporali ad alta frequenza nei propri modelli". Consiglio di leggere l'eccellente post di Dan nella sua interezza: nn to cpp: What you need to know about porting deep learning models to the edge [Tutto ciò che devi sapere sul porting dei modelli di deep learning verso l'edge].

Riferimenti

Abadi, Martin, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, e Li Zhang. 2016. «Deep Learning with Differential Privacy». In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 308–18. CCS '16. New York, NY, USA: ACM. https://doi.org/10.1145/2976749.2978318.

Abdelkader, Ahmed, Michael J. Curry, Liam Fowl, Tom Goldstein, Avi Schwarzschild, Manli Shu, Christoph Studer, e Chen Zhu. 2020. «Headless Horseman: Adversarial Attacks on Transfer Learning Models». In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 3087–91. IEEE. https://doi.org/10.1109/icassp40776.2020.9053181.

Addepalli, Sravanti, B. S. Vivek, Arya Baburaj, Gaurang Sriramanan, e R. Venkatesh Babu. 2020. «Towards Achieving Adversarial Robustness by Enforcing Feature Consistency Across Bit Planes». In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 1020–29. IEEE. https://doi.org/10.1109/cvpr42600.2020.00110.

Adolf, Robert, Saketh Rama, Brandon Reagen, Gu-yeon Wei, e David Brooks. 2016. «Fathom: Reference workloads for modern deep learning methods». In *2016 IEEE International Symposium on Workload Characterization (IISWC)*, 1–10. IEEE; IEEE. https://doi.org/10.1109/iiswc.2016.7581275.

Agarwal, Alekh, Alina Beygelzimer, Miroslav Dudík, John Langford, e Hanna M. Wallach. 2018. «A Reductions Approach to Fair Classification». In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, a cura di Jennifer G. Dy e Andreas Krause, 80:60–69. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v80/agarwal18a.html.

Agnesina, Anthony, Puranjay Rajvanshi, Tian Yang, Geraldo Pradipta, Austin Jiao, Ben Keller, Brucek Khailany, e Haoxing Ren. 2023. «AutoDMP: Automated DREAMPlace-based Macro Placement». In *Proceedings of the 2023 International Symposium on Physical Design*, 149–57. ACM. https://doi.org/10.1145/3569052.3578923.

Agrawal, Dakshi, Selcuk Baktir, Deniz Karakoyunlu, Pankaj Rohatgi, e Berk Sunar. 2007. «Trojan Detection using IC Fingerprinting». In *2007 IEEE Symposium on Security and Privacy (SP '07)*, 29–45. Springer; IEEE. https://doi.org/10.1109/sp.2007.36.

Ahmadilivani, Mohammad Hasan, Mahdi Taheri, Jaan Raik, Masoud Daneshtalab, e Maksim Jenihhin. 2024. «A Systematic Literature Review on Hardware Reliability Assessment Methods for Deep Neural Networks». *ACM Comput. Surv.* 56 (6): 1–39. https://doi.org/10.1145/3638242.

Aledhari, Mohammed, Rehma Razzak, Reza M. Parizi, e Fahad Saeed. 2020. «Federated Learning: A Survey on Enabling Technologies, Protocols, and Applications». *#IEEE_O_ACC#* 8: 140699–725. https://doi.org/10.1109/access.2020.3013541.

Alghamdi, Wael, Hsiang Hsu, Haewon Jeong, Hao Wang, Peter Michalak, Shahab Asoodeh, e Flavio Calmon. 2022. «Beyond Adult and COMPAS: Fair multi-class prediction via information projection». *Adv. Neur. In.* 35: 38747–60.

Altayeb, Moez, Marco Zennaro, e Marcelo Rovai. 2022. «Classifying mosquito wingbeat sound using TinyML». In *Proceedings of the 2022 ACM Conference on Information Technology for Social Good*, 132–37. ACM. https://doi.org/10.1145/3524458.3547258.

Amiel, Frederic, Christophe Clavier, e Michael Tunstall. 2006. «Fault analysis of DPA-resistant algorithms». In *International Workshop on Fault Diagnosis and Tolerance in Cryptography*, 223–36. Springer.

Ansel, Jason, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, et al. 2024. «PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation». In *Proceedings of the 29th ACM*

International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2, a cura di Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, e Roman Garnett, 8024–35. ACM. https://doi.org/10.1145/3620665.3640366.
Anthony, Lasse F. Wolff, Benjamin Kanding, e Raghavendra Selvan. 2020. ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems.
Antol, Stanislaw, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, e Devi Parikh. 2015. «VQA: Visual Question Answering». In *2015 IEEE International Conference on Computer Vision (ICCV)*, 2425–33. IEEE. https://doi.org/10.1109/iccv.2015.279.
Antonakakis, Manos, Tim April, Michael Bailey, Matt Bernhard, Elie Bursztein, Jaime Cochran, Zakir Durumeric, et al. 2017. «Understanding the mirai botnet». In *26th USENIX security symposium (USENIX Security 17)*, 1093–1110.
Ardila, Rosana, Megan Branson, Kelly Davis, Michael Kohler, Josh Meyer, Michael Henretty, Reuben Morais, Lindsay Saunders, Francis Tyers, e Gregor Weber. 2020. «Common Voice: A Massively-Multilingual Speech Corpus». In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, 4218–22. Marseille, France: European Language Resources Association. https://aclanthology.org/2020.lrec-1.520.
Arifeen, Tooba, Abdus Sami Hassan, e Jeong-A Lee. 2020. «Approximate Triple Modular Redundancy: A Survey». *#IEEE_O_ACC#* 8: 139851–67. https://doi.org/10.1109/access.2020.3012673.
Asonov, D., e R. Agrawal. 2004. «Keyboard acoustic emanations». In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, 3–11. IEEE; IEEE. https://doi.org/10.1109/secpri.2004.1301311.
Ateniese, Giuseppe, Luigi V. Mancini, Angelo Spognardi, Antonio Villani, Domenico Vitali, e Giovanni Felici. 2015. «Hacking smart machines with smarter ones: How to extract meaningful data from machine learning classifiers». *Int. J. Secur. Netw.* 10 (3): 137. https://doi.org/10.1504/ijsn.2015.071829.
Attia, Zachi I., Alan Sugrue, Samuel J. Asirvatham, Michael J. Ackerman, Suraj Kapa, Paul A. Friedman, e Peter A. Noseworthy. 2018. «Noninvasive assessment of dofetilide plasma concentration using a deep learning (neural network) analysis of the surface electrocardiogram: A proof of concept study». *PLOS ONE* 13 (8): e0201059. https://doi.org/10.1371/journal.pone.0201059.
Aygun, Sercan, Ece Olcay Gunes, e Christophe De Vleeschouwer. 2021. «Efficient and robust bitstream processing in binarised neural networks». *Electron. Lett.* 57 (5): 219–22. https://doi.org/10.1049/ell2.12045.
Bai, Tao, Jinqi Luo, Jun Zhao, Bihan Wen, e Qian Wang. 2021. «Recent advances in adversarial training for adversarial robustness». *arXiv preprint arXiv:2102.01356*.
Bains, Sunny. 2020. «The business of building brains». *Nature Electronics* 3 (7): 348–51. https://doi.org/10.1038/s41928-020-0449-1.
Bamoumen, Hatim, Anas Temouden, Nabil Benamar, e Yousra Chtouki. 2022. «How TinyML Can be Leveraged to Solve Environmental Problems: A Survey». In *2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT)*, 338–43. IEEE; IEEE. https://doi.org/10.1109/3ict56508.2022.9990661.
Bank, Dor, Noam Koenigstein, e Raja Giryes. 2023. «Autoencoders». *Machine Learning for Data Science Handbook: Data Mining and Knowledge Discovery Handbook*, 353–74.
Bannon, Pete, Ganesh Venkataramanan, Debjit Das Sarma, e Emil Talpes. 2019. «Computer and Redundancy Solution for the Full Self-Driving Computer». In *2019 IEEE Hot Chips 31 Symposium (HCS)*, 1–22. IEEE Computer Society; IEEE. https://doi.org/10.1109/hotchips.2019.8875645.

Barenghi, Alessandro, Guido M. Bertoni, Luca Breveglieri, Mauro Pellicioli, e Gerardo Pelosi. 2010. «Low voltage fault attacks to AES». In *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 7–12. IEEE; IEEE. https://doi.org/10.1109/hst.2010.5513121.

Barroso, Luiz André, Urs Hölzle, e Parthasarathy Ranganathan. 2019. *The Datacenter as a Computer: Designing Warehouse-Scale Machines*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01761-2.

Bau, David, Bolei Zhou, Aditya Khosla, Aude Oliva, e Antonio Torralba. 2017. «Network Dissection: Quantifying Interpretability of Deep Visual Representations». In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 3319–27. IEEE. https://doi.org/10.1109/cvpr.2017.354.

Beaton, Albert E., e John W. Tukey. 1974. «The Fitting of Power Series, Meaning Polynomials, Illustrated on Band-Spectroscopic Data». *Technometrics* 16 (2): 147. https://doi.org/10.2307/1267936.

Beck, Nathaniel, e Simon Jackman. 1998. «Beyond Linearity by Default: Generalized Additive Models». *Am. J. Polit. Sci.* 42 (2): 596. https://doi.org/10.2307/2991772.

Bender, Emily M., e Batya Friedman. 2018. «Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science». *Transactions of the Association for Computational Linguistics* 6 (dicembre): 587–604. https://doi.org/10.1162/tacl_a_00041.

Berger, Vance W, e YanYan Zhou. 2014. «Kolmogorovsmirnov test: Overview». *Wiley statsref: Statistics reference online*.

Beyer, Lucas, Olivier J Hénaff, Alexander Kolesnikov, Xiaohua Zhai, e Aäron van den Oord. 2020. «Are we done with imagenet?» *ArXiv preprint* abs/2006.07159. https://arxiv.org/abs/2006.07159.

Bhagoji, Arjun Nitin, Warren He, Bo Li, e Dawn Song. 2018. «Practical black-box attacks on deep neural networks using efficient query mechanisms». In *Proceedings of the European conference on computer vision (ECCV)*, 154–69.

Bhardwaj, Kshitij, Marton Havasi, Yuan Yao, David M. Brooks, José Miguel Hernández-Lobato, e Gu-Yeon Wei. 2020. «A comprehensive methodology to determine optimal coherence interfaces for many-accelerator SoCs». In *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*, 145–50. ACM. https://doi.org/10.1145/3370748.3406564.

Bianco, Simone, Remi Cadene, Luigi Celona, e Paolo Napoletano. 2018. «Benchmark analysis of representative deep neural network architectures». *IEEE access* 6: 64270–77.

Biega, Asia J., Peter Potash, Hal Daumé, Fernando Diaz, e Michèle Finck. 2020. «Operationalizing the Legal Principle of Data Minimization for Personalization». In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, a cura di Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, e Yiqun Liu, 399–408. ACM. https://doi.org/10.1145/3397271.3401034.

Biggio, Battista, Blaine Nelson, e Pavel Laskov. 2012. «Poisoning Attacks against Support Vector Machines». In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress. http://icml.cc/2012/papers/880.pdf.

Biggs, John, James Myers, Jedrzej Kufel, Emre Ozer, Simon Craske, Antony Sou, Catherine Ramsdale, Ken Williamson, Richard Price, e Scott White. 2021. «A natively flexible 32-bit Arm microprocessor». *Nature* 595 (7868): 532–36. https://doi.org/10.1038/s41586-021-03625-w.

Binkert, Nathan, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, et al. 2011. «The gem5 simulator». *ACM SIGARCH Computer Architecture News* 39 (2): 1–7. https://doi.org/10.1145/2024716.2024718.
Bohr, Adam, e Kaveh Memarzadeh. 2020. «The rise of artificial intelligence in healthcare applications». In *Artificial Intelligence in Healthcare*, 25–60. Elsevier. https://doi.org/10.1016/b978-0-12-818438-7.00002-2.
Bolchini, Cristiana, Luca Cassano, Antonio Miele, e Alessandro Toschi. 2023. «Fast and Accurate Error Simulation for CNNs Against Soft Errors». *IEEE Trans. Comput.* 72 (4): 984–97. https://doi.org/10.1109/tc.2022.3184274.
Bondi, Elizabeth, Ashish Kapoor, Debadeepta Dey, James Piavis, Shital Shah, Robert Hannaford, Arvind Iyer, Lucas Joppa, e Milind Tambe. 2018. «Near Real-Time Detection of Poachers from Drones in AirSim». In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, a cura di Jérôme Lang, 5814–16. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2018/847.
Bourtoule, Lucas, Varun Chandrasekaran, Christopher A. Choquette-Choo, Hengrui Jia, Adelin Travers, Baiwu Zhang, David Lie, e Nicolas Papernot. 2021. «Machine Unlearning». In *2021 IEEE Symposium on Security and Privacy (SP)*, 141–59. IEEE; IEEE. https://doi.org/10.1109/sp40001.2021.00019.
Breier, Jakub, Xiaolu Hou, Dirmanto Jap, Lei Ma, Shivam Bhasin, e Yang Liu. 2018. «Deeplaser: Practical fault attack on deep neural networks». *ArXiv preprint* abs/1806.05859. https://arxiv.org/abs/1806.05859.
Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, et al. 2020. «Language Models are Few-Shot Learners». In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, a cura di Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, e Hsuan-Tien Lin. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.
Buolamwini, Joy, e Timnit Gebru. 2018. «Gender shades: Intersectional accuracy disparities in commercial gender classification». In *Conference on fairness, accountability and transparency*, 77–91. PMLR.
Burnet, David, e Richard Thomas. 1989. «Spycatcher: The Commodification of Truth». *J. Law Soc.* 16 (2): 210. https://doi.org/10.2307/1410360.
Burr, Geoffrey W., Matthew J. BrightSky, Abu Sebastian, Huai-Yu Cheng, Jau-Yi Wu, Sangbum Kim, Norma E. Sosa, et al. 2016. «Recent Progress in Phase-Change?Pub _newline ?Memory Technology». *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 6 (2): 146–62. https://doi.org/10.1109/jetcas.2016.2547718.
Bushnell, Michael L, e Vishwani D Agrawal. 2002. «Built-in self-test». *Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits*, 489–548.
Buyya, Rajkumar, Anton Beloglazov, e Jemal Abawajy. 2010. «Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges». https://arxiv.org/abs/1006.0308.
Cai, Carrie J., Emily Reif, Narayan Hegde, Jason Hipp, Been Kim, Daniel Smilkov, Martin Wattenberg, et al. 2019. «Human-Centered Tools for Coping with Imperfect Algorithms During Medical Decision-Making». In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, a cura di Jennifer G. Dy e Andreas Krause, 80:2673–82. Proceedings of Machine Learning Research. ACM. https://doi.org/10.1145/3290605.3300234.
Cai, Han, Chuang Gan, Ligeng Zhu, e Song Han 0003. 2020. «TinyTL: Reduce Memory, Not Parameters for Efficient On-Device Learning.» In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS*

2020, December 6-12, 2020, virtual, a cura di Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, e Hsuan-Tien Lin. https://proceedings.neurips.cc/paper/2020/hash/81f7acabd411274fcf65ce2070ed568a-Abstract.html.

Cai, Han, Ligeng Zhu, e Song Han. 2019. «ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware». In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=HylVB3AqYm.

Calvo, Rafael A, Dorian Peters, Karina Vold, e Richard M Ryan. 2020. «Supporting human autonomy in AI systems: A framework for ethical enquiry». *Ethics of digital well-being: A multidisciplinary approach*, 31–54.

Carlini, Nicholas, Pratyush Mishra, Tavish Vaidya, Yuankai Zhang, Micah Sherr, Clay Shields, David Wagner, e Wenchao Zhou. 2016. «Hidden voice commands». In *25th USENIX security symposium (USENIX security 16)*, 513–30.

Carlini, Nicolas, Jamie Hayes, Milad Nasr, Matthew Jagielski, Vikash Sehwag, Florian Tramer, Borja Balle, Daphne Ippolito, e Eric Wallace. 2023. «Extracting training data from diffusion models». In *32nd USENIX Security Symposium (USENIX Security 23)*, 5253–70.

Carta, Salvatore, Alessandro Sebastian Podda, Diego Reforgiato Recupero, e Roberto Saia. 2020. «A Local Feature Engineering Strategy to Improve Network Anomaly Detection». *Future Internet* 12 (10): 177. https://doi.org/10.3390/fi12100177.

Cavoukian, Ann. 2009. «Privacy by design». *Office of the Information and Privacy Commissioner*.

Cenci, Marcelo Pilotto, Tatiana Scarazzato, Daniel Dotto Munchen, Paula Cristina Dartora, Hugo Marcelo Veit, Andrea Moura Bernardes, e Pablo R. Dias. 2021. «Eco-Friendly ElectronicsA Comprehensive Review». *Adv. Mater. Technol.* 7 (2): 2001263. https://doi.org/10.1002/admt.202001263.

Challenge, WEF Net-Zero. 2021. «The Supply Chain Opportunity». In *World Economic Forum: Geneva, Switzerland*.

Chandola, Varun, Arindam Banerjee, e Vipin Kumar. 2009. «Anomaly detection: A survey». *ACM Comput. Surv.* 41 (3): 1–58. https://doi.org/10.1145/1541880.1541882.

Chapelle, O., B. Scholkopf, e A. Zien Eds. 2009. «Semi-Supervised Learning (Chapelle, O. et al., Eds.; 2006) [Book reviews]». *IEEE Trans. Neural Networks* 20 (3): 542–42. https://doi.org/10.1109/tnn.2009.2015974.

Chen, Chaofan, Oscar Li, Daniel Tao, Alina Barnett, Cynthia Rudin, e Jonathan Su. 2019. «This Looks Like That: Deep Learning for Interpretable Image Recognition». In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, a cura di Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, e Roman Garnett, 8928–39. https://proceedings.neurips.cc/paper/2019/hash/adf7ee2dcf142b0e11888e72b43fcb75-Abstract.html.

Chen, Emma, Shvetank Prakash, Vijay Janapa Reddi, David Kim, e Pranav Rajpurkar. 2023. «A framework for integrating artificial intelligence for clinical care with continuous therapeutic monitoring». *Nature Biomedical Engineering*, novembre. https://doi.org/10.1038/s41551-023-01115-0.

Chen, H.-W. 2006. «Gallium, Indium, and Arsenic Pollution of Groundwater from a Semiconductor Manufacturing Area of Taiwan». *B. Environ. Contam. Tox.* 77 (2): 289–96. https://doi.org/10.1007/s00128-006-1062-3.

Chen, Tianqi, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, et al. 2018. «TVM: An automated End-to-End optimizing compiler for deep learning».

In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 578–94.
Chen, Tianqi, Bing Xu, Chiyuan Zhang, e Carlos Guestrin. 2016. «Training Deep Nets with Sublinear Memory Cost». *ArXiv preprint* abs/1604.06174 (aprile). http://arxiv.org/abs/1604.06174v2.
Chen, Zhiyong, e Shugong Xu. 2023. «Learning domain-heterogeneous speaker recognition systems with personalized continual federated learning». *EURASIP Journal on Audio, Speech, and Music Processing* 2023 (1): 33. https://doi.org/10.1186/s13636-023-00299-2.
Chen, Zitao, Guanpeng Li, Karthik Pattabiraman, e Nathan DeBardeleben. 2019. «iBinFI/i: an efficient fault injector for safety-critical machine learning systems». In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. SC '19. New York, NY, USA: ACM. https://doi.org/10.1145/3295500.3356177.
Chen, Zitao, Niranjhana Narayanan, Bo Fang, Guanpeng Li, Karthik Pattabiraman, e Nathan DeBardeleben. 2020. «TensorFI: A Flexible Fault Injection Framework for TensorFlow Applications». In *2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE)*, 426–35. IEEE; IEEE. https://doi.org/10.1109/issre5003.2020.00047.
Cheng, Eric, Shahrzad Mirkhani, Lukasz G. Szafaryn, Chen-Yong Cher, Hyungmin Cho, Kevin Skadron, Mircea R. Stan, et al. 2016. «Clear: uC/u ross u-L/u ayer uE/u xploration for uA/u rchitecting uR/u esilience - Combining hardware and software techniques to tolerate soft errors in processor cores». In *Proceedings of the 53rd Annual Design Automation Conference*, 1–6. ACM. https://doi.org/10.1145/2897937.2897996.
Cheng, Yu, Duo Wang, Pan Zhou, e Tao Zhang. 2018. «Model Compression and Acceleration for Deep Neural Networks: The Principles, Progress, and Challenges». *IEEE Signal Process Mag.* 35 (1): 126–36. https://doi.org/10.1109/msp.2017.2765695.
Chi, Ping, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, e Yuan Xie. 2016. «Prime: a novel processing-in-memory architecture for neural network computation in ReRAM-based main memory». *ACM SIGARCH Computer Architecture News* 44 (3): 27–39. https://doi.org/10.1145/3007787.3001140.
Chollet, François. 2018. «Introduction to keras». *March 9th*.
Christiano, Paul F., Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, e Dario Amodei. 2017. «Deep Reinforcement Learning from Human Preferences». In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, a cura di Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, e Roman Garnett, 4299–4307. https://proceedings.neurips.cc/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.
Chu, Grace, Okan Arikan, Gabriel Bender, Weijun Wang, Achille Brighton, Pieter-Jan Kindermans, Hanxiao Liu, Berkin Akin, Suyog Gupta, e Andrew Howard. 2021. «Discovering Multi-Hardware Mobile Models via Architecture Search». In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 3022–31. IEEE. https://doi.org/10.1109/cvprw53098.2021.00337.
Chua, L. 1971. «Memristor-The missing circuit element». *#IEEE_J_CT#* 18 (5): 507–19. https://doi.org/10.1109/tct.1971.1083337.
Chung, Jae-Won, Yile Gu, Insu Jang, Luoxi Meng, Nikhil Bansal, e Mosharaf Chowdhury. 2023. «Perseus: Removing Energy Bloat from Large Model Training». *ArXiv preprint* abs/2312.06902. https://arxiv.org/abs/2312.06902.
Cohen, Maxime C., Ruben Lobel, e Georgia Perakis. 2016. «The Impact of Demand Uncertainty on Consumer Subsidies for Green Technology Adoption». *Manage. Sci.* 62 (5): 1235–58. https://doi.org/10.1287/mnsc.2015.2173.

Coleman, Cody, Edward Chou, Julian Katz-Samuels, Sean Culatana, Peter Bailis, Alexander C. Berg, Robert D. Nowak, Roshan Sumbaly, Matei Zaharia, e I. Zeki Yalniz. 2022. «Similarity Search for Efficient Active Learning and Search of Rare Concepts». In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, 6402–10. AAAI Press. https://ojs.aaai.org/index.php/AAAI/article/view/20591.

Coleman, Cody, Daniel Kang, Deepak Narayanan, Luigi Nardi, Tian Zhao, Jian Zhang, Peter Bailis, Kunle Olukotun, Chris Ré, e Matei Zaharia. 2019. «Analysis of DAWNBench, a Time-to-Accuracy Machine Learning Performance Benchmark». *ACM SIGOPS Operating Systems Review* 53 (1): 14–25. https://doi.org/10.1145/3352020.3352024.

Constantinescu, Cristian. 2008. «Intermittent faults and effects on reliability of integrated circuits». In *2008 Annual Reliability and Maintainability Symposium*, 370–74. IEEE; IEEE. https://doi.org/10.1109/rams.2008.4925824.

Cooper, Tom, Suzanne Fallender, Joyann Pafumi, Jon Dettling, Sebastien Humbert, e Lindsay Lessard. 2011. «A semiconductor company's examination of its water footprint approach». In *Proceedings of the 2011 IEEE International Symposium on Sustainable Systems and Technology*, 1–6. IEEE; IEEE. https://doi.org/10.1109/issst.2011.5936865.

Cope, Gord. 2009. «Pure water, semiconductors and the recession». *Global Water Intelligence* 10 (10).

Courbariaux, Matthieu, Itay Hubara, Daniel Soudry, Ran El-Yaniv, e Yoshua Bengio. 2016. «Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1». *arXiv preprint arXiv:1602.02830*.

Crankshaw, Daniel, Xin Wang, Guilio Zhou, Michael J Franklin, Joseph E Gonzalez, e Ion Stoica. 2017. «Clipper: A {Low-Latency} online prediction serving system». In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*, 613–27.

D'ignazio, Catherine, e Lauren F Klein. 2023. *Data feminism*. MIT press.

Darvish Rouhani, Bita, Azalia Mirhoseini, e Farinaz Koushanfar. 2017. «TinyDL: Just-in-time deep learning solution for constrained embedded systems». In *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, 1–4. IEEE. https://doi.org/10.1109/iscas.2017.8050343.

Davarzani, Samaneh, David Saucier, Purva Talegaonkar, Erin Parker, Alana Turner, Carver Middleton, Will Carroll, et al. 2023. «Closing the Wearable Gap: Footankle kinematic modeling via deep learning models based on a smart sock wearable». *Wearable Technologies* 4. https://doi.org/10.1017/wtc.2023.3.

David, Robert, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, et al. 2021. «Tensorflow lite micro: Embedded machine learning for tinyml systems». *Proceedings of Machine Learning and Systems* 3: 800–811.

Davies, Emma. 2011. «Endangered elements: Critical thinking». https://www.rsc.org/images/Endangered\%20Elements\%20-\%20Critical\%20Thinking_tcm18-196054.pdf.

Davies, Mike, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, et al. 2018. «Loihi: A Neuromorphic Manycore Processor with On-Chip Learning». *IEEE Micro* 38 (1): 82–99. https://doi.org/10.1109/mm.2018.112130359.

Davies, Mike, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, e Sumedh R. Risbud. 2021. «Advancing Neuromorphic Computing With Loihi: A Survey of Results and Outlook». *Proc. IEEE* 109 (5): 911–34. https://doi.org/10.1109/jproc.2021.3067593.

Davis, Jacqueline, Daniel Bizo, Andy Lawrence, Owen Rogers, e Max Smolaks. 2022. «Uptime Institute Global Data Center Survey 2022». Uptime Institute.

Dayarathna, Miyuru, Yonggang Wen, e Rui Fan. 2016. «Data Center Energy Consumption Modeling: A Survey». *IEEE Communications Surveys & Tutorials* 18 (1): 732–94. https://doi.org/10.1109/comst.2015.2481183.
Dean, Jeffrey, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, et al. 2012. «Large Scale Distributed Deep Networks». In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, a cura di Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, e Kilian Q. Weinberger, 1232–40. https://proceedings.neurips.cc/paper/2012/hash/6aca97005c68f1206823815f66102863-Abstract.html.
Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, e Fei-Fei Li. 2009. «ImageNet: A large-scale hierarchical image database». In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 248–55. IEEE. https://doi.org/10.1109/cvpr.2009.5206848.
Desai, Tanvi, Felix Ritchie, Richard Welpton, et al. 2016. «Five Safes: Designing data access for research». *Economics Working Paper Series* 1601: 28.
Devlin, Jacob, Ming-Wei Chang, Kenton Lee, e Kristina Toutanova. 2019. «BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding». In *Proceedings of the 2019 Conference of the North*, 4171–86. Minneapolis, Minnesota: Association for Computational Linguistics. https://doi.org/10.18653/v1/n19-1423.
Dhar, Sauptik, Junyao Guo, Jiayi (Jason) Liu, Samarth Tripathi, Unmesh Kurup, e Mohak Shah. 2021. «A Survey of On-Device Machine Learning: An Algorithms and Learning Theory Perspective». *ACM Transactions on Internet of Things* 2 (3): 1–49. https://doi.org/10.1145/3450494.
Dong, Xin, Barbara De Salvo, Meng Li, Chiao Liu, Zhongnan Qu, H. T. Kung, e Ziyun Li. 2022. «SplitNets: Designing Neural Architectures for Efficient Distributed Computing on Head-Mounted Systems». In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12549–59. IEEE. https://doi.org/10.1109/cvpr52688.2022.01223.
Dongarra, Jack J. 2009. «The evolution of high performance computing on system z». *IBM J. Res. Dev.* 53: 3–4.
Duarte, Javier, Nhan Tran, Ben Hawks, Christian Herwig, Jules Muhizi, Shvetank Prakash, e Vijay Janapa Reddi. 2022. «FastML Science Benchmarks: Accelerating Real-Time Scientific Edge Machine Learning». *ArXiv preprint* abs/2207.07958. https://arxiv.org/abs/2207.07958.
Duchi, John C., Elad Hazan, e Yoram Singer. 2010. «Adaptive Subgradient Methods for Online Learning and Stochastic Optimization». In *COLT 2010 - The 23rd Conference on Learning Theory, Haifa, Israel, June 27-29, 2010*, a cura di Adam Tauman Kalai e Mehryar Mohri, 257–69. Omnipress. http://colt2010.haifa.il.ibm.com/papers/COLT2010proceedings.pdf#page=265.
Duisterhof, Bardienus P, Srivatsan Krishnan, Jonathan J Cruz, Colby R Banbury, William Fu, Aleksandra Faust, Guido CHE de Croon, e Vijay Janapa Reddi. 2019. «Learning to seek: Autonomous source seeking with deep reinforcement learning onboard a nano drone microcontroller». *ArXiv preprint* abs/1909.11236. https://arxiv.org/abs/1909.11236.
Duisterhof, Bardienus P., Shushuai Li, Javier Burgues, Vijay Janapa Reddi, e Guido C. H. E. de Croon. 2021. «Sniffy Bug: A Fully Autonomous Swarm of Gas-Seeking Nano Quadcopters in Cluttered Environments». In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 9099–9106. IEEE; IEEE. https://doi.org/10.1109/iros51168.2021.9636217.
Dürr, Marc, Gunnar Nissen, Kurt-Wolfram Sühs, Philipp Schwenkenbecher, Christian Geis, Marius Ringelstein, Hans-Peter Hartung, et al. 2021. «CSF Findings in Acute NMDAR and LGI1

Antibody–Associated Autoimmune Encephalitis». *Neurology Neuroimmunology & Neuroinflammation* 8 (6). https://doi.org/10.1212/nxi.0000000000001086.
Dwork, Cynthia, Frank McSherry, Kobbi Nissim, e Adam Smith. 2006. «Calibrating Noise to Sensitivity in Private Data Analysis». In *Theory of Cryptography*, a cura di Shai Halevi e Tal Rabin, 265–84. Berlin, Heidelberg: Springer Berlin Heidelberg.
Dwork, Cynthia, e Aaron Roth. 2013. «The Algorithmic Foundations of Differential Privacy». *Foundations and Trends® in Theoretical Computer Science* 9 (3-4): 211–407. https://doi.org/10.1561/0400000042.
Ebrahimi, Khosrow, Gerard F. Jones, e Amy S. Fleischer. 2014. «A review of data center cooling technology, operating conditions and the corresponding low-grade waste heat recovery opportunities». *Renewable Sustainable Energy Rev.* 31 (marzo): 622–38. https://doi.org/10.1016/j.rser.2013.12.007.
Egwutuoha, Ifeanyi P., David Levy, Bran Selic, e Shiping Chen. 2013. «A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems». *The Journal of Supercomputing* 65 (3): 1302–26. https://doi.org/10.1007/s11227-013-0884-0.
Eisenman, Assaf, Kiran Kumar Matam, Steven Ingram, Dheevatsa Mudigere, Raghuraman Krishnamoorthi, Krishnakumar Nair, Misha Smelyanskiy, e Murali Annavaram. 2022. «Check-N-Run: A checkpointing system for training deep learning recommendation models». In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, 929–43.
Eldan, Ronen, e Mark Russinovich. 2023. «Who's Harry Potter? Approximate Unlearning in LLMs». *ArXiv preprint* abs/2310.02238. https://arxiv.org/abs/2310.02238.
El-Rayis, A. O. 2014. «Reconfigurable architectures for the next generation of mobile device telecommunications systems». : https://www.researchgate.net/publication/292608967.
Eshraghian, Jason K., Max Ward, Emre O. Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, e Wei D. Lu. 2023. «Training Spiking Neural Networks Using Lessons From Deep Learning». *Proc. IEEE* 111 (9): 1016–54. https://doi.org/10.1109/jproc.2023.3308088.
Esteva, Andre, Brett Kuprel, Roberto A. Novoa, Justin Ko, Susan M. Swetter, Helen M. Blau, e Sebastian Thrun. 2017. «Dermatologist-level classification of skin cancer with deep neural networks». *Nature* 542 (7639): 115–18. https://doi.org/10.1038/nature21056.
«EuroSoil 2021 (O205)». 2021. In *EuroSoil 2021 (O205)*. DS12902. STMicroelectronics; Frontiers Media SA. https://doi.org/10.3389/978-2-88966-997-4.
Eykholt, Kevin, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, e Dawn Song. 2017. «Robust Physical-World Attacks on Deep Learning Models». *ArXiv preprint* abs/1707.08945. https://arxiv.org/abs/1707.08945.
Fahim, Farah, Benjamin Hawks, Christian Herwig, James Hirschauer, Sergo Jindariani, Nhan Tran, Luca P. Carloni, et al. 2021. «hls4ml: An Open-Source Codesign Workflow to Empower Scientific Low-Power Machine Learning Devices». https://arxiv.org/abs/2103.05579.
Farah, Martha J. 2005. «Neuroethics: The practical and the philosophical». *Trends Cogn. Sci.* 9 (1): 34–40. https://doi.org/10.1016/j.tics.2004.12.001.
Farwell, James P., e Rafal Rohozinski. 2011. «Stuxnet and the Future of Cyber War». *Survival* 53 (1): 23–40. https://doi.org/10.1080/00396338.2011.555586.
Fowers, Jeremy, Kalin Ovtcharov, Michael Papamichael, Todd Massengill, Ming Liu, Daniel Lo, Shlomi Alkalay, et al. 2018. «A Configurable Cloud-Scale DNN Processor for Real-Time AI». In *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*, 1–14. IEEE; IEEE. https://doi.org/10.1109/isca.2018.00012.
Francalanza, Adrian, Luca Aceto, Antonis Achilleos, Duncan Paul Attard, Ian Cassar, Dario Della Monica, e Anna Ingólfsdóttir. 2017. «A foundation for runtime monitoring». In *International Conference on Runtime Verification*, 8–29. Springer.

Frankle, Jonathan, e Michael Carbin. 2019. «The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks». In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net. https://openreview.net/forum?id=rJl-b3RcF7.
Friedman, Batya. 1996. «Value-sensitive design». *Interactions* 3 (6): 16–23. https://doi.org/10.1145/242485.242493.
Furber, Steve. 2016. «Large-scale neuromorphic computing systems». *J. Neural Eng.* 13 (5): 051001. https://doi.org/10.1088/1741-2560/13/5/051001.
Fursov, Ivan, Matvey Morozov, Nina Kaploukhaya, Elizaveta Kovtun, Rodrigo Rivera-Castro, Gleb Gusev, Dmitry Babaev, Ivan Kireev, Alexey Zaytsev, e Evgeny Burnaev. 2021. «Adversarial Attacks on Deep Models for Financial Transaction Records». In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2868–78. ACM. https://doi.org/10.1145/3447548.3467145.
Gale, Trevor, Erich Elsen, e Sara Hooker. 2019. «The state of sparsity in deep neural networks». *ArXiv preprint* abs/1902.09574. https://arxiv.org/abs/1902.09574.
Gandolfi, Karine, Christophe Mourtel, e Francis Olivier. 2001. «Electromagnetic analysis: Concrete results». In *Cryptographic Hardware and Embedded SystemsCHES 2001: Third International Workshop Paris, France, May 1416, 2001 Proceedings 3*, 251–61. Springer.
Gannot, G., e M. Ligthart. 1994. «Verilog HDL based FPGA design». In *International Verilog HDL Conference*, 86–92. IEEE. https://doi.org/10.1109/ivc.1994.323743.
Gao, Yansong, Said F. Al-Sarawi, e Derek Abbott. 2020. «Physical unclonable functions». *Nature Electronics* 3 (2): 81–91. https://doi.org/10.1038/s41928-020-0372-5.
Gates, Byron D. 2009. «Flexible Electronics». *Science* 323 (5921): 1566–67. https://doi.org/10.1126/science.1171230.
Gebru, Timnit, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III, e Kate Crawford. 2021. «Datasheets for datasets». *Commun. ACM* 64 (12): 86–92. https://doi.org/10.1145/3458723.
Geiger, Atticus, Hanson Lu, Thomas Icard, e Christopher Potts. 2021. «Causal Abstractions of Neural Networks». In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, a cura di Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, e Jennifer Wortman Vaughan, 9574–86. https://proceedings.neurips.cc/paper/2021/hash/4f5c422f4d49a5a807eda27434231040-Abstract.html.
Gholami, Dong Kim, Mahoney Yao, e Keutzer. 2021. «A Survey of Quantization Methods for Efficient Neural Network Inference)». *ArXiv preprint*. https://arxiv.org/abs/2103.13630.
Glorot, Xavier, e Yoshua Bengio. 2010. «Understanding the difficulty of training deep feedforward neural networks.» In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 249–56. http://proceedings.mlr.press/v9/glorot10a.html.
Gnad, Dennis R. E., Fabian Oboril, e Mehdi B. Tahoori. 2017. «Voltage drop-based fault attacks on FPGAs using valid bitstreams». In *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*, 1–7. IEEE; IEEE. https://doi.org/10.23919/fpl.2017.8056840.
Goodfellow, Ian, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, e Yoshua Bengio. 2020. «Generative adversarial networks». *Commun. ACM* 63 (11): 139–44. https://doi.org/10.1145/3422622.
Goodyear, Victoria A. 2017. «Social media, apps and wearable technologies: Navigating ethical dilemmas and procedures». *Qualitative Research in Sport, Exercise and Health* 9 (3): 285–302. https://doi.org/10.1080/2159676x.2017.1303790.
Google. s.d. «Information quality content moderation». https://blog.google/documents/83/.

Gordon, Ariel, Elad Eban, Ofir Nachum, Bo Chen, Hao Wu, Tien-Ju Yang, e Edward Choi. 2018. «MorphNet: Fast & Simple Resource-Constrained Structure Learning of Deep Networks». In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1586–95. IEEE. https://doi.org/10.1109/cvpr.2018.00171.
Gräfe, Ralf, Qutub Syed Sha, Florian Geissler, e Michael Paulitsch. 2023. «Large-Scale Application of Fault Injection into PyTorch Models -an Extension to PyTorchFI for Validation Efficiency». In *2023 53rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*, 56–62. IEEE; IEEE. https://doi.org/10.1109/dsn-s58398.2023.00025.
Greengard, Samuel. 2015. *The Internet of Things*. The MIT Press. https://doi.org/10.7551/mitpress/10277.001.0001.
Grossman, Elizabeth. 2007. *High tech trash: Digital devices, hidden toxics, and human health*. Island press.
Gruslys, Audrunas, Rémi Munos, Ivo Danihelka, Marc Lanctot, e Alex Graves. 2016. «Memory-Efficient Backpropagation Through Time». In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, a cura di Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, e Roman Garnett, 4125–33. https://proceedings.neurips.cc/paper/2016/hash/a501bebf79d570651ff601788ea9d16d-Abstract.html.
Gu, Ivy. 2023. «Deep Learning Model Compression (ii) by Ivy Gu Medium». https://ivygdy.medium.com/deep-learning-model-compression-ii-546352ea9453.
Gujarati, Arpan, Reza Karimi, Safya Alzayat, Wei Hao, Antoine Kaufmann, Ymir Vigfusson, e Jonathan Mace. 2020. «Serving DNNs like Clockwork: Performance Predictability from the Bottom Up.» In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 443–62. https://www.usenix.org/conference/osdi20/presentation/gujarati.
Guo, Chuan, Jacob Gardner, Yurong You, Andrew Gordon Wilson, e Kilian Weinberger. 2019. «Simple black-box adversarial attacks». In *International conference on machine learning*, 2484–93. PMLR.
Guo, Yutao, Hao Wang, Hui Zhang, Tong Liu, Zhaoguang Liang, Yunlong Xia, Li Yan, et al. 2019. «Mobile Photoplethysmographic Technology to Detect Atrial Fibrillation». *Journal of the American College of Cardiology* 74 (19): 2365–75. https://doi.org/10.1016/j.jacc.2019.08.019.
Gupta, Maanak, Charankumar Akiri, Kshitiz Aryal, Eli Parker, e Lopamudra Praharaj. 2023. «From ChatGPT to ThreatGPT: Impact of Generative AI in Cybersecurity and Privacy». *#IEEE_O_ACC#* 11: 80218–45. https://doi.org/10.1109/access.2023.3300381.
Gupta, Maya, Andrew Cotter, Jan Pfeifer, Konstantin Voevodski, Kevin Canini, Alexander Mangylov, Wojciech Moczydlowski, e Alexander Van Esbroeck. 2016. «Monotonic calibrated interpolated look-up tables». *The Journal of Machine Learning Research* 17 (1): 3790–3836.
Gupta, Udit, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, e Carole-Jean Wu. 2022. «Act: designing sustainable computer systems with an architectural carbon modeling tool». In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, 784–99. ACM. https://doi.org/10.1145/3470496.3527408.
Gwennap, Linley. s.d. «Certus-NX Innovates General-Purpose FPGAs».
Haensch, Wilfried, Tayfun Gokmen, e Ruchir Puri. 2019. «The Next Generation of Deep Learning Hardware: Analog Computing». *Proc. IEEE* 107 (1): 108–22. https://doi.org/10.1109/jproc.2018.2871057.
Hamming, R. W. 1950. «Error Detecting and Error Correcting Codes». *Bell Syst. Tech. J.* 29 (2): 147–60. https://doi.org/10.1002/j.1538-7305.1950.tb00463.x.

Han, Song, Huizi Mao, e William J Dally. 2015. «Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding». *arXiv preprint arXiv:1510.00149*.
Han, Song, Huizi Mao, e William J. Dally. 2016. «Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding». https://arxiv.org/abs/1510.00149.
Handlin, Oscar. 1965. «Science and technology in popular culture». *Daedalus-us.*, 156–70.
Hardt, Moritz, Eric Price, e Nati Srebro. 2016. «Equality of Opportunity in Supervised Learning». In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, a cura di Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, e Roman Garnett, 3315–23. https://proceedings.neurips.cc/paper/2016/hash/9d2682367c3935defcb1f9e247a97c0d-Abstract.html.
Hawks, Benjamin, Javier Duarte, Nicholas J. Fraser, Alessandro Pappalardo, Nhan Tran, e Yaman Umuroglu. 2021. «Ps and Qs: Quantization-aware Pruning for Efficient Low Latency Neural Network Inference». *Frontiers in Artificial Intelligence* 4 (luglio). https://doi.org/10.3389/frai.2021.676564.
Hazan, Avi, e Elishai Ezra Tsur. 2021. «Neuromorphic Analog Implementation of Neural Engineering Framework-Inspired Spiking Neuron for High-Dimensional Representation». *Front. Neurosci.* 15 (febbraio): 627221. https://doi.org/10.3389/fnins.2021.627221.
He, Kaiming, Xiangyu Zhang, Shaoqing Ren, e Jian Sun. 2015. «Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification». In *2015 IEEE International Conference on Computer Vision (ICCV)*, 1026–34. IEEE. https://doi.org/10.1109/iccv.2015.123.
———. 2016. «Deep Residual Learning for Image Recognition». In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–78. IEEE. https://doi.org/10.1109/cvpr.2016.90.
He, Yi, Prasanna Balaprakash, e Yanjing Li. 2020. «FIdelity: Efficient Resilience Analysis Framework for Deep Learning Accelerators». In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 270–81. IEEE; IEEE. https://doi.org/10.1109/micro50266.2020.00033.
He, Yi, Mike Hutton, Steven Chan, Robert De Gruijl, Rama Govindaraju, Nishant Patil, e Yanjing Li. 2023. «Understanding and Mitigating Hardware Failures in Deep Learning Training Systems». In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 1–16. IEEE; ACM. https://doi.org/10.1145/3579371.3589105.
Hébert-Johnson, Úrsula, Michael P. Kim, Omer Reingold, e Guy N. Rothblum. 2018. «Multicalibration: Calibration for the (Computationally-Identifiable) Masses». In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, a cura di Jennifer G. Dy e Andreas Krause, 80:1944–53. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v80/hebert-johnson18a.html.
Hegde, Sumant. 2023. «An Introduction to Separable Convolutions - Analytics Vidhya». https://www.analyticsvidhya.com/blog/2021/11/an-introduction-to-separable-convolutions/.
Henderson, Peter, Jieru Hu, Joshua Romoff, Emma Brunskill, Dan Jurafsky, e Joelle Pineau. 2020. «Towards the systematic reporting of the energy and carbon footprints of machine learning». *The Journal of Machine Learning Research* 21 (1): 10039–81.
Hendrycks, Dan, e Thomas Dietterich. 2019. «Benchmarking neural network robustness to common corruptions and perturbations». *arXiv preprint arXiv:1903.12261*.

Hendrycks, Dan, Kevin Zhao, Steven Basart, Jacob Steinhardt, e Dawn Song. 2021. «Natural Adversarial Examples». In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 15262–71. IEEE. https://doi.org/10.1109/cvpr46437.2021.01501.
Hennessy, John L., e David A. Patterson. 2019. «A new golden age for computer architecture». *Commun. ACM* 62 (2): 48–60. https://doi.org/10.1145/3282307.
Himmelstein, Gracie, David Bates, e Li Zhou. 2022. «Examination of Stigmatizing Language in the Electronic Health Record». *JAMA Network Open* 5 (1): e2144967. https://doi.org/10.1001/jamanetworkopen.2021.44967.
Hinton, Geoffrey. 2005. «Van Nostrand's Scientific Encyclopedia». Wiley. https://doi.org/10.1002/0471743984.vse0673.
———. 2017. «Overview of Minibatch Gradient Descent». University of Toronto; University Lecture.
Ho Yoon, Jung, Hyung-Suk Jung, Min Hwan Lee, Gun Hwan Kim, Seul Ji Song, Jun Yeong Seok, Kyung Jean Yoon, et al. 2012. «Frontiers in Electronic Materials». Wiley. https://doi.org/10.1002/9783527667703.ch67.
Hoefler, Torsten, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, e Alexandra Peste. 2021. «Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks», gennaio. http://arxiv.org/abs/2102.00554v1.
Holland, Sarah, Ahmed Hosny, Sarah Newman, Joshua Joseph, e Kasia Chmielinski. 2020. «The Dataset Nutrition Label: A Framework to Drive Higher Data Quality Standards». In *Data Protection and Privacy*. Hart Publishing. https://doi.org/10.5040/9781509932771.ch-001.
Hong, Sanghyun, Nicholas Carlini, e Alexey Kurakin. 2023. «Publishing Efficient On-device Models Increases Adversarial Vulnerability». In *2023 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, abs 1603 5279:271–90. IEEE; IEEE. https://doi.org/10.1109/satml54575.2023.00026.
Hooker, Sara. 2021. «The hardware lottery». *Commun. ACM* 64 (12): 58–65. https://doi.org/10.1145/3467017.
Hosseini, Hossein, Sreeram Kannan, Baosen Zhang, e Radha Poovendran. 2017. «Deceiving google's perspective api built for detecting toxic comments». *ArXiv preprint* abs/1702.08138. https://arxiv.org/abs/1702.08138.
Howard, Andrew G., Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, e Hartwig Adam. 2017. «MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications». *ArXiv preprint*. https://arxiv.org/abs/1704.04861.
Hsiao, Yu-Shun, Zishen Wan, Tianyu Jia, Radhika Ghosal, Abdulrahman Mahmoud, Arijit Raychowdhury, David Brooks, Gu-Yeon Wei, e Vijay Janapa Reddi. 2023. «MAVFI: An End-to-End Fault Analysis Framework with Anomaly Detection and Recovery for Micro Aerial Vehicles». In *2023 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1–6. IEEE; IEEE. https://doi.org/10.23919/date56975.2023.10137246.
Hsu, Liang-Ching, Ching-Yi Huang, Yen-Hsun Chuang, Ho-Wen Chen, Ya-Ting Chan, Heng Yi Teah, Tsan-Yao Chen, Chiung-Fen Chang, Yu-Ting Liu, e Yu-Min Tzou. 2016. «Accumulation of heavy metals and trace elements in fluvial sediments received effluents from traditional and semiconductor industries». *Scientific Reports* 6 (1): 34250. https://doi.org/10.1038/srep34250.
Hu, Jie, Li Shen, e Gang Sun. 2018. «Squeeze-and-Excitation Networks». In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 7132–41. IEEE. https://doi.org/10.1109/cvpr.2018.00745.
Hu, Yang, Jie Jiang, Lifu Zhang, Yunfeng Shi, e Jian Shi. 2023. «Halide Perovskite Semiconductors». Wiley. https://doi.org/10.1002/9783527829026.ch13.

Huang, Tsung-Ching, Kenjiro Fukuda, Chun-Ming Lo, Yung-Hui Yeh, Tsuyoshi Sekitani, Takao Someya, e Kwang-Ting Cheng. 2011. «Pseudo-CMOS: A Design Style for Low-Cost and Robust Flexible Electronics». *IEEE Trans. Electron Devices* 58 (1): 141–50. https://doi.org/10.1109/ted.2010.2088127.
Hutter, Michael, Jorn-Marc Schmidt, e Thomas Plos. 2009. «Contact-based fault injections and power analysis on RFID tags». In *2009 European Conference on Circuit Theory and Design*, 409–12. IEEE; IEEE. https://doi.org/10.1109/ecctd.2009.5275012.
Iandola, Forrest N, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, e Kurt Keutzer. 2016. «SqueezeNet: Alexnet-level accuracy with 50x fewer parameters and 0.5 MB model size». *ArXiv preprint* abs/1602.07360. https://arxiv.org/abs/1602.07360.
Ignatov, Andrey, Radu Timofte, William Chou, Ke Wang, Max Wu, Tim Hartley, e Luc Van Gool. 2018. «AI Benchmark: Running deep neural networks on Android smartphones», 0–0.
Imani, Mohsen, Abbas Rahimi, e Tajana S. Rosing. 2016. «Resistive Configurable Associative Memory for Approximate Computing». In *Proceedings of the 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, 1327–32. IEEE; Research Publishing Services. https://doi.org/10.3850/9783981537079_0454.
IntelLabs. 2023. «Knowledge Distillation - Neural Network Distiller». https://intellabs.github.io/distiller/knowledge_distillation.html.
Ippolito, Daphne, Florian Tramer, Milad Nasr, Chiyuan Zhang, Matthew Jagielski, Katherine Lee, Christopher Choquette Choo, e Nicholas Carlini. 2023. «Preventing Generation of Verbatim Memorization in Language Models Gives a False Sense of Privacy». In *Proceedings of the 16th International Natural Language Generation Conference*, 5253–70. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.inlg-main.3.
Irimia-Vladu, Mihai. 2014. «"Green" electronics: Biodegradable and biocompatible materials and devices for sustainable future». *Chem. Soc. Rev.* 43 (2): 588–610. https://doi.org/10.1039/c3cs60235d.
Isscc. 2014. «Computing's energy problem (and what we can do about it)». https://ieeexplore.ieee.org/document/6757323.
Jacob, Benoit, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, e Dmitry Kalenichenko. 2018. «Quantization and training of neural networks for efficient integer-arithmetic-only inference». In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2704–13.
Jaderberg, Max, Valentin Dalibard, Simon Osindero, Wojciech M. Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, et al. 2017. «Population Based Training of Neural Networks». *arXiv preprint arXiv:1711.09846*, novembre. http://arxiv.org/abs/1711.09846v2.
Janapa Reddi, Vijay, Alexander Elium, Shawn Hymel, David Tischler, Daniel Situnayake, Carl Ward, Louis Moreau, et al. 2023. «Edge Impulse: An MLOps Platform for Tiny Machine Learning». *Proceedings of Machine Learning and Systems* 5.
Jha, A. R. 2014. *Rare Earth Materials: Properties and Applications*. CRC Press. https://doi.org/10.1201/b17045.
Jha, Saurabh, Subho Banerjee, Timothy Tsai, Siva K. S. Hari, Michael B. Sullivan, Zbigniew T. Kalbarczyk, Stephen W. Keckler, e Ravishankar K. Iyer. 2019. «ML-Based Fault Injection for Autonomous Vehicles: A Case for Bayesian Fault Injection». In *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 112–24. IEEE; IEEE. https://doi.org/10.1109/dsn.2019.00025.
Jia, Yangqing, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, e Trevor Darrell. 2014. «Caffe: Convolutional Architecture for Fast Feature Embedding». In *Proceedings of the 22nd ACM international conference on Multimedia*, 675–78. ACM. https://doi.org/10.1145/2647868.2654889.

Jia, Zhe, Marco Maggioni, Benjamin Staiger, e Daniele P. Scarpazza. 2018. «Dissecting the NVIDIA Volta GPU Architecture via Microbenchmarking». *ArXiv preprint*. https://arxiv.org/abs/1804.06826.
Jia, Zhenge, Dawei Li, Xiaowei Xu, Na Li, Feng Hong, Lichuan Ping, e Yiyu Shi. 2023. «Life-threatening ventricular arrhythmia detection challenge in implantable cardioverterdefibrillators». *Nature Machine Intelligence* 5 (5): 554–55. https://doi.org/10.1038/s42256-023-00659-9.
Jia, Zhihao, Matei Zaharia, e Alex Aiken. 2019. «Beyond Data and Model Parallelism for Deep Neural Networks». In *Proceedings of Machine Learning and Systems 2019, MLSys 2019, Stanford, CA, USA, March 31 - April 2, 2019*, a cura di Ameet Talwalkar, Virginia Smith, e Matei Zaharia. mlsys.org. https://proceedings.mlsys.org/book/265.pdf.
Jin, Yilun, Xiguang Wei, Yang Liu, e Qiang Yang. 2020. «Towards utilizing unlabeled data in federated learning: A survey and prospective». *arXiv preprint arXiv:2002.11545*.
Johnson-Roberson, Matthew, Charles Barto, Rounak Mehta, Sharath Nittur Sridhar, Karl Rosaen, e Ram Vasudevan. 2017. «Driving in the Matrix: Can virtual worlds replace human-generated annotations for real world tasks?» In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 746–53. Singapore, Singapore: IEEE. https://doi.org/10.1109/icra.2017.7989092.
Jouppi, Norman P., Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, et al. 2017a. «In-Datacenter Performance Analysis of a Tensor Processing Unit». In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12. ISCA '17. New York, NY, USA: ACM. https://doi.org/10.1145/3079856.3080246.
———, et al. 2017b. «In-Datacenter Performance Analysis of a Tensor Processing Unit». In *Proceedings of the 44th Annual International Symposium on Computer Architecture*, 1–12. ISCA '17. New York, NY, USA: ACM. https://doi.org/10.1145/3079856.3080246.
Jouppi, Norm, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, et al. 2023. «TPU v4: An Optically Reconfigurable Supercomputer for Machine Learning with Hardware Support for Embeddings». In *Proceedings of the 50th Annual International Symposium on Computer Architecture*. ISCA '23. New York, NY, USA: ACM. https://doi.org/10.1145/3579371.3589350.
Joye, Marc, e Michael Tunstall. 2012. *Fault Analysis in Cryptography*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-29656-7.
Kairouz, Peter, Sewoong Oh, e Pramod Viswanath. 2015. «Secure Multi-party Differential Privacy.» In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, a cura di Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, e Roman Garnett, 2008–16. https://proceedings.neurips.cc/paper/2015/hash/a01610228fe998f515a72dd730294d87-Abstract.html.
Kalamkar, Dhiraj, Dheevatsa Mudigere, Naveen Mellempudi, Dipankar Das, Kunal Banerjee, Sasikanth Avancha, Dharma Teja Vooturi, et al. 2019. «A Study of BFLOAT16 for Deep Learning Training». https://arxiv.org/abs/1905.12322.
Kao, Sheng-Chun, Geonhwa Jeong, e Tushar Krishna. 2020. «ConfuciuX: Autonomous Hardware Resource Assignment for DNN Accelerators using Reinforcement Learning». In *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 622–36. IEEE; IEEE. https://doi.org/10.1109/micro50266.2020.00058.
Kao, Sheng-Chun, e Tushar Krishna. 2020. «Gamma: automating the HW mapping of DNN models on accelerators via genetic algorithm». In *Proceedings of the 39th International*

Conference on Computer-Aided Design, 1–9. ACM. https://doi.org/10.1145/3400302.3415639.
Kaplan, Jared, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, e Dario Amodei. 2020. «Scaling Laws for Neural Language Models». *ArXiv preprint* abs/2001.08361. https://arxiv.org/abs/2001.08361.
Karargyris, Alexandros, Renato Umeton, Micah J. Sheller, Alejandro Aristizabal, Johnu George, Anna Wuest, Sarthak Pati, et al. 2023. «Federated benchmarking of medical artificial intelligence with MedPerf». *Nature Machine Intelligence* 5 (7): 799–810. https://doi.org/10.1038/s42256-023-00652-2.
Kaur, Harmanpreet, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, e Jennifer Wortman Vaughan. 2020. «Interpreting Interpretability: Understanding Data Scientists' Use of Interpretability Tools for Machine Learning». In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, a cura di Regina Bernhaupt, Florian 'Floyd'Mueller, David Verweij, Josh Andres, Joanna McGrenere, Andy Cockburn, Ignacio Avellino, et al., 1–14. ACM. https://doi.org/10.1145/3313831.3376219.
Kawazoe Aguilera, Marcos, Wei Chen, e Sam Toueg. 1997. «Heartbeat: A timeout-free failure detector for quiescent reliable communication». In *Distributed Algorithms: 11th International Workshop, WDAG'97 Saarbrücken, Germany, September 2426, 1997 Proceedings 11*, 126–40. Springer.
Khan, Mohammad Emtiyaz, e Siddharth Swaroop. 2021. «Knowledge-Adaptation Priors». In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, a cura di Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, e Jennifer Wortman Vaughan, 19757–70. https://proceedings.neurips.cc/paper/2021/hash/a4380923dd651c195b1631af7c829187-Abstract.html.
Kiela, Douwe, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, et al. 2021. «Dynabench: Rethinking Benchmarking in NLP». In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 4110–24. Online: Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.naacl-main.324.
Kim, Jungrae, Michael Sullivan, e Mattan Erez. 2015. «Bamboo ECC: Strong, safe, and flexible codes for reliable computer memory». In *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, 101–12. IEEE; IEEE. https://doi.org/10.1109/hpca.2015.7056025.
Kim, Sunju, Chungsik Yoon, Seunghon Ham, Jihoon Park, Ohun Kwon, Donguk Park, Sangjun Choi, Seungwon Kim, Kwonchul Ha, e Won Kim. 2018. «Chemical use in the semiconductor manufacturing industry». *Int. J. Occup. Env. Heal.* 24 (3-4): 109–18. https://doi.org/10.1080/10773525.2018.1519957.
Kingma, Diederik P., e Jimmy Ba. 2014. «Adam: A Method for Stochastic Optimization». A cura di Yoshua Bengio e Yann LeCun, dicembre. http://arxiv.org/abs/1412.6980v9.
Kirkpatrick, James, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, et al. 2017. «Overcoming catastrophic forgetting in neural networks». *Proc. Natl. Acad. Sci.* 114 (13): 3521–26. https://doi.org/10.1073/pnas.1611835114.
Ko, Yohan. 2021. «Characterizing System-Level Masking Effects against Soft Errors». *Electronics* 10 (18): 2286. https://doi.org/10.3390/electronics10182286.
Kocher, Paul, Jann Horn, Anders Fogh, Daniel Genkin, Daniel Gruss, Werner Haas, Mike Hamburg, et al. 2019b. «Spectre Attacks: Exploiting Speculative Execution». In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. https://doi.org/10.1109/sp.2019.00002.

———, et al. 2019a. «Spectre Attacks: Exploiting Speculative Execution». In *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE. https://doi.org/10.1109/sp.2019.00002.
Kocher, Paul, Joshua Jaffe, e Benjamin Jun. 1999. «Differential power analysis». In *Advances in CryptologyCRYPTO'99: 19th Annual International Cryptology Conference Santa Barbara, California, USA, August 1519, 1999 Proceedings 19*, 388–97. Springer.
Kocher, Paul, Joshua Jaffe, Benjamin Jun, e Pankaj Rohatgi. 2011. «Introduction to differential power analysis». *Journal of Cryptographic Engineering* 1 (1): 5–27. https://doi.org/10.1007/s13389-011-0006-y.
Koh, Pang Wei, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, e Percy Liang. 2020. «Concept Bottleneck Models». In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, 119:5338–48. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v119/koh20a.html.
Koh, Pang Wei, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, et al. 2021. «WILDS: A Benchmark of in-the-Wild Distribution Shifts». In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, a cura di Marina Meila e Tong Zhang, 139:5637–64. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v139/koh21a.html.
Koren, Yehuda, Robert Bell, e Chris Volinsky. 2009. «Matrix Factorization Techniques for Recommender Systems». *Computer* 42 (8): 30–37. https://doi.org/10.1109/mc.2009.263.
Krishna, Adithya, Srikanth Rohit Nudurupati, Chandana D G, Pritesh Dwivedi, André van Schaik, Mahesh Mehendale, e Chetan Singh Thakur. 2023. «RAMAN: A Re-configurable and Sparse TinyML Accelerator for Inference on Edge». https://arxiv.org/abs/2306.06493.
Krishnamoorthi. 2018. «Quantizing deep convolutional networks for efficient inference: A whitepaper». *ArXiv preprint*. https://arxiv.org/abs/1806.08342.
Krishnan, Rayan, Pranav Rajpurkar, e Eric J. Topol. 2022. «Self-supervised learning in medicine and healthcare». *Nat. Biomed. Eng.* 6 (12): 1346–52. https://doi.org/10.1038/s41551-022-00914-1.
Krishnan, Srivatsan, Natasha Jaques, Shayegan Omidshafiei, Dan Zhang, Izzeddin Gur, Vijay Janapa Reddi, e Aleksandra Faust. 2022. «Multi-Agent Reinforcement Learning for Microprocessor Design Space Exploration». https://arxiv.org/abs/2211.16385.
Krishnan, Srivatsan, Amir Yazdanbakhsh, Shvetank Prakash, Jason Jabbour, Ikechukwu Uchendu, Susobhan Ghosh, Behzad Boroujerdian, et al. 2023. «ArchGym: An Open-Source Gymnasium for Machine Learning Assisted Architecture Design». In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, 1–16. ACM. https://doi.org/10.1145/3579371.3589049.
Krizhevsky, Alex, Ilya Sutskever, e Geoffrey E. Hinton. 2012. «ImageNet Classification with Deep Convolutional Neural Networks». In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, a cura di Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, e Kilian Q. Weinberger, 1106–14. https://proceedings.neurips.cc/paper/2012/hash/c399862d3b9d6b76c8436e924a68c45b-Abstract.html.
———. 2017. «ImageNet classification with deep convolutional neural networks». A cura di F. Pereira, C. J. Burges, L. Bottou, e K. Q. Weinberger. *Commun. ACM* 60 (6): 84–90. https://doi.org/10.1145/3065386.
Kung, Hsiang Tsung, e Charles E Leiserson. 1979. «Systolic arrays (for VLSI)». In *Sparse Matrix Proceedings 1978*, 1:256–82. Society for industrial; applied mathematics Philadelphia, PA, USA.

Kurth, Thorsten, Shashank Subramanian, Peter Harrington, Jaideep Pathak, Morteza Mardani, David Hall, Andrea Miele, Karthik Kashinath, e Anima Anandkumar. 2023. «FourCastNet: Accelerating Global High-Resolution Weather Forecasting Using Adaptive Fourier Neural Operators». In *Proceedings of the Platform for Advanced Scientific Computing Conference*, 1–11. ACM. https://doi.org/10.1145/3592979.3593412.
Kuzmin, Andrey, Mart Van Baalen, Yuwei Ren, Markus Nagel, Jorn Peters, e Tijmen Blankevoort. 2022. «FP8 Quantization: The Power of the Exponent». https://arxiv.org/abs/2208.09225.
Kuznetsova, Alina, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, et al. 2020. «The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale». *International journal of computer vision* 128 (7): 1956–81.
Kwon, Jisu, e Daejin Park. 2021. «Hardware/Software Co-Design for TinyML Voice-Recognition Application on Resource Frugal Edge Devices». *Applied Sciences* 11 (22): 11073. https://doi.org/10.3390/app112211073.
Kwon, Sun Hwa, e Lin Dong. 2022. «Flexible sensors and machine learning for heart monitoring». *Nano Energy* 102 (novembre): 107632. https://doi.org/10.1016/j.nanoen.2022.107632.
Kwon, Young D., Rui Li, Stylianos I. Venieris, Jagmohan Chauhan, Nicholas D. Lane, e Cecilia Mascolo. 2023. «TinyTrain: Resource-Aware Task-Adaptive Sparse Training of DNNs at the Data-Scarce Edge». *ArXiv preprint* abs/2307.09988 (luglio). http://arxiv.org/abs/2307.09988v2.
Lai, Liangzhen, Naveen Suda, e Vikas Chandra. 2018a. «Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus». *ArXiv preprint* abs/1801.06601. https://arxiv.org/abs/1801.06601.
———. 2018b. «CMSIS-NN: Efficient Neural Network Kernels for Arm Cortex-M CPUs». https://arxiv.org/abs/1801.06601.
Lakkaraju, Himabindu, e Osbert Bastani. 2020. «"How do I fool you?": Manipulating User Trust via Misleading Black Box Explanations». In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, 79–85. ACM. https://doi.org/10.1145/3375627.3375833.
Lam, Remi, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, et al. 2023. «Learning skillful medium-range global weather forecasting». *Science* 382 (6677): 1416–21. https://doi.org/10.1126/science.adi2336.
Lannelongue, Loïc, Jason Grealey, e Michael Inouye. 2021. «Green Algorithms: Quantifying the Carbon Footprint of Computation». *Adv. Sci.* 8 (12): 2100707. https://doi.org/10.1002/advs.202100707.
LeCun, Yann, John Denker, e Sara Solla. 1989. «Optimal brain damage». *Adv Neural Inf Process Syst* 2.
Lee, Minwoong, Namho Lee, Huijeong Gwon, Jongyeol Kim, Younggwan Hwang, e Seongik Cho. 2022. «Design of Radiation-Tolerant High-Speed Signal Processing Circuit for Detecting Prompt Gamma Rays by Nuclear Explosion». *Electronics* 11 (18): 2970. https://doi.org/10.3390/electronics11182970.
LeRoy Poff, N, MM Brinson, e JW Day. 2002. «Aquatic ecosystems & Global climate change». *Pew Center on Global Climate Change*.
Li, En, Liekang Zeng, Zhi Zhou, e Xu Chen. 2020. «Edge AI: On-demand Accelerating Deep Neural Network Inference via Edge Computing». *IEEE Trans. Wireless Commun.* 19 (1): 447–57. https://doi.org/10.1109/twc.2019.2946140.
Li, Guanpeng, Siva Kumar Sastry Hari, Michael Sullivan, Timothy Tsai, Karthik Pattabiraman, Joel Emer, e Stephen W. Keckler. 2017. «Understanding error propagation in deep learning neural network (DNN) accelerators and applications». In *Proceedings of the International*

Conference for High Performance Computing, Networking, Storage and Analysis, 1–12. ACM. https://doi.org/10.1145/3126908.3126964.

Li, Jingzhen, Igbe Tobore, Yuhang Liu, Abhishek Kandwal, Lei Wang, e Zedong Nie. 2021. «Non-invasive Monitoring of Three Glucose Ranges Based On ECG By Using DBSCAN-CNN». *IEEE Journal of Biomedical and Health Informatics* 25 (9): 3340–50. https://doi.org/10.1109/jbhi.2021.3072628.

Li, Mu, David G. Andersen, Alexander J. Smola, e Kai Yu. 2014. «Communication Efficient Distributed Machine Learning with the Parameter Server». In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, a cura di Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, e Kilian Q. Weinberger, 19–27. https://proceedings.neurips.cc/paper/2014/hash/1ff1de774005f8da13f42943881c655f-Abstract.html.

Li, Qinbin, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, e Bingsheng He. 2023. «A Survey on Federated Learning Systems: Vision, Hype and Reality for Data Privacy and Protection». *IEEE Trans. Knowl. Data Eng.* 35 (4): 3347–66. https://doi.org/10.1109/tkde.2021.3124599.

Li, Tian, Anit Kumar Sahu, Ameet Talwalkar, e Virginia Smith. 2020. «Federated Learning: Challenges, Methods, and Future Directions». *IEEE Signal Process Mag.* 37 (3): 50–60. https://doi.org/10.1109/msp.2020.2975749.

Li, Xiang, Tao Qin, Jian Yang, e Tie-Yan Liu. 2016. «LightRNN: Memory and Computation-Efficient Recurrent Neural Networks». In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, a cura di Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, e Roman Garnett, 4385–93. https://proceedings.neurips.cc/paper/2016/hash/c3e4035af2a1cde9f21e1ae1951ac80b-Abstract.html.

Li, Yuhang, Xin Dong, e Wei Wang. 2020. «Additive Powers-of-Two Quantization: An Efficient Non-uniform Discretization for Neural Networks». In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=BkgXT24tDS.

Li, Zhizhong, e Derek Hoiem. 2018. «Learning without Forgetting». *IEEE Trans. Pattern Anal. Mach. Intell.* 40 (12): 2935–47. https://doi.org/10.1109/tpami.2017.2773081.

Li, Zhuohan, Lianmin Zheng, Yinmin Zhong, Vincent Liu, Ying Sheng, Xin Jin, Yanping Huang, et al. 2023. «{AlpaServe}: Statistical multiplexing with model parallelism for deep learning serving». In *17th USENIX Symposium on Operating Systems Design and Implementation (OSDI 23)*, 663–79.

Lin, Ji, Wei-Ming Chen, Yujun Lin, John Cohn, Chuang Gan, e Song Han. 2020. «MCUNet: Tiny Deep Learning on IoT Devices». In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, a cura di Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, e Hsuan-Tien Lin. https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html.

Lin, Ji, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, e Song Han. 2022. «On-device training under 256kb memory». *Adv. Neur. In.* 35: 22941–54.

Lin, Ji, Ligeng Zhu, Wei-Ming Chen, Wei-Chen Wang, e Song Han. 2023. «Tiny Machine Learning: Progress and Futures Feature». *IEEE Circuits Syst. Mag.* 23 (3): 8–34. https://doi.org/10.1109/mcas.2023.3302182.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, e C Lawrence Zitnick. 2014. «Microsoft coco: Common objects in context». In *Computer VisionECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–55. Springer.
Lindgren, Simon. 2023. *Handbook of Critical Studies of Artificial Intelligence*. Edward Elgar Publishing.
Lindholm, Andreas, Dave Zachariah, Petre Stoica, e Thomas B. Schon. 2019. «Data Consistency Approach to Model Validation». *#IEEE_O_ACC#* 7: 59788–96. https://doi.org/10.1109/access.2019.2915109.
Lindholm, Erik, John Nickolls, Stuart Oberman, e John Montrym. 2008. «NVIDIA Tesla: A Unified Graphics and Computing Architecture». *IEEE Micro* 28 (2): 39–55. https://doi.org/10.1109/mm.2008.31.
Lin, Tang Tang, Dang Yang, e Han Gan. 2023. «AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration». *ArXiv preprint*. https://arxiv.org/abs/2306.00978.
Liu, Yanan, Xiaoxia Wei, Jinyu Xiao, Zhijie Liu, Yang Xu, e Yun Tian. 2020. «Energy consumption and emission mitigation prediction based on data center traffic and PUE for global data centers». *Global Energy Interconnection* 3 (3): 272–82. https://doi.org/10.1016/j.gloei.2020.07.008.
Liu, Yingcheng, Guo Zhang, Christopher G. Tarolli, Rumen Hristov, Stella Jensen-Roberts, Emma M. Waddell, Taylor L. Myers, et al. 2022. «Monitoring gait at home with radio waves in Parkinson's disease: A marker of severity, progression, and medication response». *Science Translational Medicine* 14 (663): eadc9669. https://doi.org/10.1126/scitranslmed.adc9669.
Loh, Gabriel H. 2008. «3D-Stacked Memory Architectures for Multi-core Processors». *ACM SIGARCH Computer Architecture News* 36 (3): 453–64. https://doi.org/10.1145/1394608.1382159.
Lopez-Paz, David, e Marc'Aurelio Ranzato. 2017. «Gradient episodic memory for continual learning». *Adv Neural Inf Process Syst* 30.
Lou, Yin, Rich Caruana, Johannes Gehrke, e Giles Hooker. 2013. «Accurate intelligible models with pairwise interactions». In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, a cura di Inderjit S. Dhillon, Yehuda Koren, Rayid Ghani, Ted E. Senator, Paul Bradley, Rajesh Parekh, Jingrui He, Robert L. Grossman, e Ramasamy Uthurusamy, 623–31. ACM. https://doi.org/10.1145/2487575.2487579.
Lowy, Andrew, Rakesh Pavan, Sina Baharlouei, Meisam Razaviyayn, e Ahmad Beirami. 2021. «Fermi: Fair empirical risk minimization via exponential Rényi mutual information».
Lubana, Ekdeep Singh, e Robert P Dick. 2020. «A gradient flow framework for analyzing network pruning». *arXiv preprint arXiv:2009.11839*.
Luebke, David. 2008. «CUDA: Scalable parallel programming for high-performance scientific computing». In *2008 5th IEEE International Symposium on Biomedical Imaging: From Nano to Macro*, 836–38. IEEE. https://doi.org/10.1109/isbi.2008.4541126.
Lundberg, Scott M., e Su-In Lee. 2017. «A Unified Approach to Interpreting Model Predictions». In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, a cura di Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, e Roman Garnett, 4765–74. https://proceedings.neurips.cc/paper/2017/hash/8a20a8621978632d76c43dfd28b67767-Abstract.html.
Ma, Dongning, Fred Lin, Alban Desmaison, Joel Coburn, Daniel Moore, Sriram Sankar, e Xun Jiao. 2024. «Dr. DNA: Combating Silent Data Corruptions in Deep Learning using Distribution of Neuron Activations». In *Proceedings of the 29th ACM International Conference on*

Architectural Support for Programming Languages and Operating Systems, Volume 3, 239–52. ACM. https://doi.org/10.1145/3620666.3651349.
Maas, Martin, David G. Andersen, Michael Isard, Mohammad Mahdi Javanmard, Kathryn S. McKinley, e Colin Raffel. 2024. «Combining Machine Learning and Lifetime-Based Resource Management for Memory Allocation and Beyond». *Commun. ACM* 67 (4): 87–96. https://doi.org/10.1145/3611018.
Maass, Wolfgang. 1997. «Networks of spiking neurons: The third generation of neural network models». *Neural Networks* 10 (9): 1659–71. https://doi.org/10.1016/s0893-6080(97)00011-7.
Madry, Aleksander, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, e Adrian Vladu. 2017. «Towards deep learning models resistant to adversarial attacks». *arXiv preprint arXiv:1706.06083*.
Mahmoud, Abdulrahman, Neeraj Aggarwal, Alex Nobbe, Jose Rodrigo Sanchez Vicarte, Sarita V. Adve, Christopher W. Fletcher, Iuri Frosio, e Siva Kumar Sastry Hari. 2020. «PyTorchFI: A Runtime Perturbation Tool for DNNs». In *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*, 25–31. IEEE; IEEE. https://doi.org/10.1109/dsn-w50199.2020.00014.
Mahmoud, Abdulrahman, Siva Kumar Sastry Hari, Christopher W. Fletcher, Sarita V. Adve, Charbel Sakr, Naresh Shanbhag, Pavlo Molchanov, Michael B. Sullivan, Timothy Tsai, e Stephen W. Keckler. 2021. «Optimizing Selective Protection for CNN Resilience». In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*, 127–38. IEEE. https://doi.org/10.1109/issre52982.2021.00025.
Mahmoud, Abdulrahman, Thierry Tambe, Tarek Aloui, David Brooks, e Gu-Yeon Wei. 2022. «GoldenEye: A Platform for Evaluating Emerging Numerical Data Formats in DNN Accelerators». In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 206–14. IEEE. https://doi.org/10.1109/dsn53405.2022.00031.
Marković, Danijela, Alice Mizrahi, Damien Querlioz, e Julie Grollier. 2020. «Physics for neuromorphic computing». *Nature Reviews Physics* 2 (9): 499–510. https://doi.org/10.1038/s42254-020-0208-2.
Martin, C. Dianne. 1993. «The myth of the awesome thinking machine». *Commun. ACM* 36 (4): 120–33. https://doi.org/10.1145/255950.153587.
Marulli, Fiammetta, Stefano Marrone, e Laura Verde. 2022. «Sensitivity of Machine Learning Approaches to Fake and Untrusted Data in Healthcare Domain». *Journal of Sensor and Actuator Networks* 11 (2): 21. https://doi.org/10.3390/jsan11020021.
Maslej, Nestor, Loredana Fattorini, Erik Brynjolfsson, John Etchemendy, Katrina Ligett, Terah Lyons, James Manyika, et al. 2023. «Artificial intelligence index report 2023». *ArXiv preprint* abs/2310.03715. https://arxiv.org/abs/2310.03715.
Mattson, Peter, Vijay Janapa Reddi, Christine Cheng, Cody Coleman, Greg Diamos, David Kanter, Paulius Micikevicius, et al. 2020b. «MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance». *IEEE Micro* 40 (2): 8–16. https://doi.org/10.1109/mm.2020.2974843.
———, et al. 2020a. «MLPerf: An Industry Standard Benchmark Suite for Machine Learning Performance». *IEEE Micro* 40 (2): 8–16. https://doi.org/10.1109/mm.2020.2974843.
Mazumder, Mark, Sharad Chitlangia, Colby Banbury, Yiping Kang, Juan Manuel Ciro, Keith Achorn, Daniel Galvez, et al. 2021. «Multilingual spoken words corpus». In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.
McCarthy, John. 1981. «Epistemological Problems Of Artificial Intelligence». In *Readings in Artificial Intelligence*, 459–65. Elsevier. https://doi.org/10.1016/b978-0-934613-03-3.50035-0.
McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, e Blaise Agüera y Arcas. 2017a. «Communication-Efficient Learning of Deep Networks from Decentralized Data». In

Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, AISTATS 2017, 20-22 April 2017, Fort Lauderdale, FL, USA, a cura di Aarti Singh e Xiaojin (Jerry) Zhu, 54:1273–82. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v54/mcmahan17a.html.
———. 2017b. «Communication-Efficient Learning of Deep Networks from Decentralized Data.» In *Artificial intelligence and statistics*, 1273–82. PMLR. http://proceedings.mlr.press/v54/mcmahan17a.html.
Miller, Charlie. 2019. «Lessons learned from hacking a car». *IEEE Design & Test* 36 (6): 7–9. https://doi.org/10.1109/mdat.2018.2863106.
Miller, Charlie, e Chris Valasek. 2015. «Remote exploitation of an unaltered passenger vehicle». *Black Hat USA* 2015 (S 91): 1–91.
Miller, D. A. B. 2000. «Optical interconnects to silicon». *#IEEE_J_JSTQE#* 6 (6): 1312–17. https://doi.org/10.1109/2944.902184.
Mills, Andrew, e Stephen Le Hunte. 1997. «An overview of semiconductor photocatalysis». *J. Photochem. Photobiol., A* 108 (1): 1–35. https://doi.org/10.1016/s1010-6030(97)00118-4.
Mirhoseini, Azalia, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, et al. 2021. «A graph placement methodology for fast chip design». *Nature* 594 (7862): 207–12. https://doi.org/10.1038/s41586-021-03544-w.
Mishra, Asit K., Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, e Paulius Micikevicius. 2021. «Accelerating Sparse Deep Neural Networks». *CoRR* abs/2104.08378. https://arxiv.org/abs/2104.08378.
Mittal, Sparsh, Gaurav Verma, Brajesh Kaushik, e Farooq A. Khanday. 2021. «A survey of SRAM-based in-memory computing techniques and applications». *J. Syst. Architect.* 119 (ottobre): 102276. https://doi.org/10.1016/j.sysarc.2021.102276.
Modha, Dharmendra S., Filipp Akopyan, Alexander Andreopoulos, Rathinakumar Appuswamy, John V. Arthur, Andrew S. Cassidy, Pallab Datta, et al. 2023. «Neural inference at the frontier of energy, space, and time». *Science* 382 (6668): 329–35. https://doi.org/10.1126/science.adh1174.
Mohanram, K., e N. A. Touba. 2003. «Partial error masking to reduce soft error failure rate in logic circuits». In *Proceedings. 16th IEEE Symposium on Computer Arithmetic*, 433–40. IEEE; IEEE Comput. Soc. https://doi.org/10.1109/dftvs.2003.1250141.
Monyei, Chukwuka G., e Kirsten E. H. Jenkins. 2018. «Electrons have no identity: Setting right misrepresentations in Google and Apple's clean energy purchasing». *Energy Research & Social Science* 46 (dicembre): 48–51. https://doi.org/10.1016/j.erss.2018.06.015.
Moshawrab, Mohammad, Mehdi Adda, Abdenour Bouzouane, Hussein Ibrahim, e Ali Raad. 2023. «Reviewing Federated Learning Aggregation Algorithms; Strategies, Contributions, Limitations and Future Perspectives». *Electronics* 12 (10): 2287. https://doi.org/10.3390/electronics12102287.
Mukherjee, S. S., J. Emer, e S. K. Reinhardt. 2005. «The Soft Error Problem: An Architectural Perspective». In *11th International Symposium on High-Performance Computer Architecture*, 243–47. IEEE; IEEE. https://doi.org/10.1109/hpca.2005.37.
Munshi, Aaftab. 2009. «The OpenCL specification». In *2009 IEEE Hot Chips 21 Symposium (HCS)*, 1–314. IEEE. https://doi.org/10.1109/hotchips.2009.7478342.
Musk, Elon et al. 2019. «An Integrated Brain-Machine Interface Platform With Thousands of Channels». *J. Med. Internet Res.* 21 (10): e16194. https://doi.org/10.2196/16194.
Myllyaho, Lalli, Mikko Raatikainen, Tomi Männistö, Jukka K. Nurminen, e Tommi Mikkonen. 2022. «On misbehaviour and fault tolerance in machine learning systems». *J. Syst. Software* 183 (gennaio): 111096. https://doi.org/10.1016/j.jss.2021.111096.
Nakano, Jane. 2021. *The geopolitics of critical minerals supply chains*. JSTOR.

Narayanan, Arvind, e Vitaly Shmatikov. 2006. «How to break anonymity of the netflix prize dataset». *arXiv preprint cs/0610105*.
Ng, Davy Tsz Kit, Jac Ka Lok Leung, Kai Wah Samuel Chu, e Maggie Shen Qiao. 2021. «AI literacy: Definition, teaching, evaluation and ethical issues». *Proceedings of the Association for Information Science and Technology* 58 (1): 504–9.
Ngo, Richard, Lawrence Chan, e Sören Mindermann. 2022. «The alignment problem from a deep learning perspective». *ArXiv preprint* abs/2209.00626. https://arxiv.org/abs/2209.00626.
Nguyen, Ngoc-Bao, Keshigeyan Chandrasegaran, Milad Abdollahzadeh, e Ngai-Man Cheung. 2023. «Re-Thinking Model Inversion Attacks Against Deep Neural Networks». In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 16384–93. IEEE. https://doi.org/10.1109/cvpr52729.2023.01572.
Norrie, Thomas, Nishant Patil, Doe Hyun Yoon, George Kurian, Sheng Li, James Laudon, Cliff Young, Norman Jouppi, e David Patterson. 2021. «The Design Process for Google's Training Chips: Tpuv2 and TPUv3». *IEEE Micro* 41 (2): 56–63. https://doi.org/10.1109/mm.2021.3058217.
Northcutt, Curtis G, Anish Athalye, e Jonas Mueller. 2021. «Pervasive Label Errors in Test Sets Destabilize Machine Learning Benchmarks». *arXiv*. https://doi.org/https://doi.org/10.48550/arXiv.2103.14749 arXiv-issued DOI via DataCite.
Obermeyer, Ziad, Brian Powers, Christine Vogeli, e Sendhil Mullainathan. 2019. «Dissecting racial bias in an algorithm used to manage the health of populations». *Science* 366 (6464): 447–53. https://doi.org/10.1126/science.aax2342.
Oecd. 2023. «A blueprint for building national compute capacity for artificial intelligence». 350. Organisation for Economic Co-Operation; Development (OECD). https://doi.org/10.1787/876367e3-en.
Olah, Chris, Nick Cammarata, Ludwig Schubert, Gabriel Goh, Michael Petrov, e Shan Carter. 2020. «Zoom In: An Introduction to Circuits». *Distill* 5 (3): e00024–001. https://doi.org/10.23915/distill.00024.001.
Oliynyk, Daryna, Rudolf Mayer, e Andreas Rauber. 2023. «I Know What You Trained Last Summer: A Survey on Stealing Machine Learning Models and Defences». *ACM Comput. Surv.* 55 (14s): 1–41. https://doi.org/10.1145/3595292.
Ooko, Samson Otieno, Marvin Muyonga Ogore, Jimmy Nsenga, e Marco Zennaro. 2021. «TinyML in Africa: Opportunities and Challenges». In *2021 IEEE Globecom Workshops (GC Wkshps)*, 1–6. IEEE; IEEE. https://doi.org/10.1109/gcwkshps52748.2021.9682107.
Oprea, Alina, Anoop Singhal, e Apostol Vassilev. 2022. «Poisoning Attacks Against Machine Learning: Can Machine Learning Be Trustworthy?» *Computer* 55 (11): 94–99. https://doi.org/10.1109/mc.2022.3190787.
Pan, Sinno Jialin, e Qiang Yang. 2010. «A Survey on Transfer Learning». *IEEE Transactions on Knowledge and Data Engineering* 22 (10): 1345–59. https://doi.org/10.1109/tkde.2009.191.
Panda, Priyadarshini, Indranil Chakraborty, e Kaushik Roy. 2019. «Discretization Based Solutions for Secure Machine Learning Against Adversarial Attacks». *#IEEE_O_ACC#* 7: 70157–68. https://doi.org/10.1109/access.2019.2919463.
Papadimitriou, George, e Dimitris Gizopoulos. 2021. «Demystifying the System Vulnerability Stack: Transient Fault Effects Across the Layers». In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 902–15. IEEE; IEEE. https://doi.org/10.1109/isca52012.2021.00075.
Papernot, Nicolas, Patrick McDaniel, Xi Wu, Somesh Jha, e Ananthram Swami. 2016. «Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks». In *2016 IEEE Symposium on Security and Privacy (SP)*, 582–97. IEEE; IEEE. https://doi.org/10.1109/sp.2016.41.

Parrish, Alicia, Hannah Rose Kirk, Jessica Quaye, Charvi Rastogi, Max Bartolo, Oana Inel, Juan Ciro, et al. 2023. «Adversarial Nibbler: A Data-Centric Challenge for Improving the Safety of Text-to-Image Models». *ArXiv preprint* abs/2305.14384. https://arxiv.org/abs/2305.14384.
Patterson, David A, e John L Hennessy. 2016. *Computer organization and design ARM edition: The hardware software interface*. Morgan kaufmann.
Patterson, David, Joseph Gonzalez, Urs Holzle, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, e Jeff Dean. 2022. «The Carbon Footprint of Machine Learning Training Will Plateau, Then Shrink». *Computer* 55 (7): 18–28. https://doi.org/10.1109/mc.2022.3148714.
Peters, Dorian, Rafael A. Calvo, e Richard M. Ryan. 2018. «Designing for Motivation, Engagement and Wellbeing in Digital Experience». *Front. Psychol.* 9 (maggio): 797. https://doi.org/10.3389/fpsyg.2018.00797.
Phillips, P Jonathon, Carina A Hahn, Peter C Fontana, David A Broniatowski, e Mark A Przybocki. 2020. «Four principles of explainable artificial intelligence». *Gaithersburg, Maryland* 18.
Plank, James S. 1997. «A tutorial on ReedSolomon coding for fault-tolerance in RAID-like systems». *Software: Practice and Experience* 27 (9): 995–1012.
Pont, Michael J, e Royan HL Ong. 2002. «Using watchdog timers to improve the reliability of single-processor embedded systems: Seven new patterns and a case study». In *Proceedings of the First Nordic Conference on Pattern Languages of Programs*, 159–200. Citeseer.
Prakash, Shvetank, Tim Callahan, Joseph Bushagour, Colby Banbury, Alan V. Green, Pete Warden, Tim Ansell, e Vijay Janapa Reddi. 2023. «CFU Playground: Full-stack Open-Source Framework for Tiny Machine Learning (TinyML) Acceleration on FPGAs». In *2023 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. Vol. abs/2201.01863. IEEE. https://doi.org/10.1109/ispass57527.2023.00024.
Prakash, Shvetank, Matthew Stewart, Colby Banbury, Mark Mazumder, Pete Warden, Brian Plancher, e Vijay Janapa Reddi. 2023. «Is TinyML Sustainable? Assessing the Environmental Impacts of Machine Learning on Microcontrollers». *ArXiv preprint*. https://arxiv.org/abs/2301.11899.
Psoma, Sotiria D., e Chryso Kanthou. 2023. «Wearable Insulin Biosensors for Diabetes Management: Advances and Challenges». *Biosensors* 13 (7): 719. https://doi.org/10.3390/bios13070719.
Pushkarna, Mahima, Andrew Zaldivar, e Oddur Kjartansson. 2022. «Data Cards: Purposeful and Transparent Dataset Documentation for Responsible AI». In *2022 ACM Conference on Fairness, Accountability, and Transparency*. ACM. https://doi.org/10.1145/3531146.3533231.
Putnam, Andrew, Adrian M. Caulfield, Eric S. Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, et al. 2014. «A reconfigurable fabric for accelerating large-scale datacenter services». *ACM SIGARCH Computer Architecture News* 42 (3): 13–24. https://doi.org/10.1145/2678373.2665678.
Qi, Chen, Shibo Shen, Rongpeng Li, Zhifeng Zhao, Qing Liu, Jing Liang, e Honggang Zhang. 2021. «An efficient pruning scheme of deep neural networks for Internet of Things applications». *EURASIP Journal on Advances in Signal Processing* 2021 (1): 31. https://doi.org/10.1186/s13634-021-00744-4.
Qian, Yu, Xuegong Zhou, Hao Zhou, e Lingli Wang. 2024. «An Efficient Reinforcement Learning Based Framework for Exploring Logic Synthesis». *ACM Trans. Des. Autom. Electron. Syst.* 29 (2): 1–33. https://doi.org/10.1145/3632174.
R. V., Rashmi, e Karthikeyan A. 2018. «Secure boot of Embedded Applications - A Review». In *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 291–98. IEEE. https://doi.org/10.1109/iceca.2018.8474730.

Rachwan, John, Daniel Zügner, Bertrand Charpentier, Simon Geisler, Morgane Ayle, e Stephan Günnemann. 2022. «Winning the lottery ahead of time: Efficient early network pruning». In *International Conference on Machine Learning*, 18293–309. PMLR.
Raina, Rajat, Anand Madhavan, e Andrew Y. Ng. 2009. «Large-scale deep unsupervised learning using graphics processors». In *Proceedings of the 26th Annual International Conference on Machine Learning*, a cura di Andrea Pohoreckyj Danyluk, Léon Bottou, e Michael L. Littman, 382:873–80. ACM International Conference Proceeding Series. ACM. https://doi.org/10.1145/1553374.1553486.
Ramaswamy, Vikram V., Sunnie S. Y. Kim, Ruth Fong, e Olga Russakovsky. 2023a. «Overlooked Factors in Concept-Based Explanations: Dataset Choice, Concept Learnability, and Human Capability». In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10932–41. IEEE. https://doi.org/10.1109/cvpr52729.2023.01052.
Ramaswamy, Vikram V, Sunnie SY Kim, Ruth Fong, e Olga Russakovsky. 2023b. «UFO: A unified method for controlling Understandability and Faithfulness Objectives in concept-based explanations for CNNs». *ArXiv preprint* abs/2303.15632. https://arxiv.org/abs/2303.15632.
Ramcharan, Amanda, Kelsee Baranowski, Peter McCloskey, Babuali Ahmed, James Legg, e David P. Hughes. 2017. «Deep Learning for Image-Based Cassava Disease Detection». *Front. Plant Sci.* 8 (ottobre): 1852. https://doi.org/10.3389/fpls.2017.01852.
Ramesh, Aditya, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, e Ilya Sutskever. 2021. «Zero-Shot Text-to-Image Generation». In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, a cura di Marina Meila e Tong Zhang, 139:8821–31. Proceedings of Machine Learning Research. PMLR. http://proceedings.mlr.press/v139/ramesh21a.html.
Ranganathan, Parthasarathy. 2011. «From Microprocessors to Nanostores: Rethinking Data-Centric Systems». *Computer* 44 (1): 39–48. https://doi.org/10.1109/mc.2011.18.
Rao, Ravi. 2021. «TinyML unlocks new possibilities for sustainable development technologies». *www.wevolver.com*. https://www.wevolver.com/article/tinyml-unlocks-new-possibilities-for-sustainable-development-technologies.
Rashid, Layali, Karthik Pattabiraman, e Sathish Gopalakrishnan. 2012. «Intermittent Hardware Errors Recovery: Modeling and Evaluation». In *2012 Ninth International Conference on Quantitative Evaluation of Systems*, 220–29. IEEE; IEEE. https://doi.org/10.1109/qest.2012.37.
———. 2015. «Characterizing the Impact of Intermittent Hardware Faults on Programs». *IEEE Trans. Reliab.* 64 (1): 297–310. https://doi.org/10.1109/tr.2014.2363152.
Ratner, Alex, Braden Hancock, Jared Dunnmon, Roger Goldman, e Christopher Ré. 2018. «Snorkel MeTaL: Weak Supervision for Multi-Task Learning». In *Proceedings of the Second Workshop on Data Management for End-To-End Machine Learning*. ACM. https://doi.org/10.1145/3209889.3209898.
Reagen, Brandon, Udit Gupta, Lillian Pentecost, Paul Whatmough, Sae Kyu Lee, Niamh Mulholland, David Brooks, e Gu-Yeon Wei. 2018. «Ares: A framework for quantifying the resilience of deep neural networks». In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 1–6. IEEE. https://doi.org/10.1109/dac.2018.8465834.
Reagen, Brandon, Jose Miguel Hernandez-Lobato, Robert Adolf, Michael Gelbart, Paul Whatmough, Gu-Yeon Wei, e David Brooks. 2017. «A case for efficient accelerator design space exploration via Bayesian optimization». In *2017 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, 1–6. IEEE; IEEE. https://doi.org/10.1109/islped.2017.8009208.
Reddi, Sashank J., Satyen Kale, e Sanjiv Kumar. 2019. «On the Convergence of Adam and Beyond». *arXiv preprint arXiv:1904.09237*, aprile. http://arxiv.org/abs/1904.09237v1.

Reddi, Vijay Janapa, Christine Cheng, David Kanter, Peter Mattson, Guenther Schmuelling, Carole-Jean Wu, Brian Anderson, et al. 2020. «MLPerf Inference Benchmark». In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, 446–59. IEEE; IEEE. https://doi.org/10.1109/isca45697.2020.00045.
Reddi, Vijay Janapa, e Meeta Sharma Gupta. 2013. *Resilient Architecture Design for Voltage Variation*. Springer International Publishing. https://doi.org/10.1007/978-3-031-01739-1.
Reis, G. A., J. Chang, N. Vachharajani, R. Rangan, e D. I. August. 2005. «SWIFT: Software Implemented Fault Tolerance». In *International Symposium on Code Generation and Optimization*, 243–54. IEEE; IEEE. https://doi.org/10.1109/cgo.2005.34.
Ribeiro, Marco Tulio, Sameer Singh, e Carlos Guestrin. 2016. «" Why should i trust you?" Explaining the predictions of any classifier». In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, 1135–44.
Robbins, Herbert, e Sutton Monro. 1951. «A Stochastic Approximation Method». *The Annals of Mathematical Statistics* 22 (3): 400–407. https://doi.org/10.1214/aoms/1177729586.
Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, e Bjorn Ommer. 2022. «High-Resolution Image Synthesis with Latent Diffusion Models». In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. https://doi.org/10.1109/cvpr52688.2022.01042.
Romero, Francisco, Qian Li 0027, Neeraja J. Yadwadkar, e Christos Kozyrakis. 2021. «INFaaS: Automated Model-less Inference Serving.» In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*, 397–411. https://www.usenix.org/conference/atc21/presentation/romero.
Rosa, Gustavo H. de, e João P. Papa. 2021. «A survey on text generation using generative adversarial networks». *Pattern Recogn.* 119 (novembre): 108098. https://doi.org/10.1016/j.patcog.2021.108098.
Rosenblatt, Frank. 1957. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory.
Roskies, Adina. 2002. «Neuroethics for the New Millenium». *Neuron* 35 (1): 21–23. https://doi.org/10.1016/s0896-6273(02)00763-8.
Ruder, Sebastian. 2016. «An overview of gradient descent optimization algorithms». *ArXiv preprint* abs/1609.04747 (settembre). http://arxiv.org/abs/1609.04747v2.
Rudin, Cynthia. 2019. «Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead». *Nature Machine Intelligence* 1 (5): 206–15. https://doi.org/10.1038/s42256-019-0048-x.
Rumelhart, David E., Geoffrey E. Hinton, e Ronald J. Williams. 1986. «Learning representations by back-propagating errors». *Nature* 323 (6088): 533–36. https://doi.org/10.1038/323533a0.
Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, et al. 2015. «ImageNet Large Scale Visual Recognition Challenge». *Int. J. Comput. Vision* 115 (3): 211–52. https://doi.org/10.1007/s11263-015-0816-y.
Russell, Stuart. 2021. «Human-compatible artificial intelligence». *Human-like machine intelligence*, 3–23.
Ryan, Richard M., e Edward L. Deci. 2000. «Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being.» *Am. Psychol.* 55 (1): 68–78. https://doi.org/10.1037/0003-066x.55.1.68.
Samajdar, Ananda, Yuhao Zhu, Paul Whatmough, Matthew Mattina, e Tushar Krishna. 2018. «Scale-sim: Systolic cnn accelerator simulator». *ArXiv preprint* abs/1811.02883. https://arxiv.org/abs/1811.02883.
Sambasivan, Nithya, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, e Lora M Aroyo. 2021a. «"Everyone wants to do the model work, not the data work": Data

Cascades in High-Stakes AI». In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, 1–15.
———. 2021b. «"Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI». In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM. https://doi.org/10.1145/3411764.3445518.
Sangchoolie, Behrooz, Karthik Pattabiraman, e Johan Karlsson. 2017. «One Bit is (Not) Enough: An Empirical Study of the Impact of Single and Multiple Bit-Flip Errors». In *2017 47th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 97–108. IEEE; IEEE. https://doi.org/10.1109/dsn.2017.30.
Schäfer, Mike S. 2023. «The Notorious GPT: Science communication in the age of artificial intelligence». *Journal of Science Communication* 22 (02): Y02. https://doi.org/10.22323/2.22020402.
Schizas, Nikolaos, Aristeidis Karras, Christos Karras, e Spyros Sioutas. 2022. «TinyML for Ultra-Low Power AI and Large Scale IoT Deployments: A Systematic Review». *Future Internet* 14 (12): 363. https://doi.org/10.3390/fi14120363.
Schuman, Catherine D., Shruti R. Kulkarni, Maryam Parsa, J. Parker Mitchell, Prasanna Date, e Bill Kay. 2022. «Opportunities for neuromorphic computing algorithms and applications». *Nature Computational Science* 2 (1): 10–19. https://doi.org/10.1038/s43588-021-00184-y.
Schwartz, Daniel, Jonathan Michael Gomes Selman, Peter Wrege, e Andreas Paepcke. 2021. «Deployment of Embedded Edge-AI for Wildlife Monitoring in Remote Regions». In *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 1035–42. IEEE; IEEE. https://doi.org/10.1109/icmla52953.2021.00170.
Schwartz, Roy, Jesse Dodge, Noah A. Smith, e Oren Etzioni. 2020. «Green AI». *Commun. ACM* 63 (12): 54–63. https://doi.org/10.1145/3381831.
Segal, Mark, e Kurt Akeley. 1999. «The OpenGL graphics system: A specification (version 1.1)».
Segura Anaya, L. H., Abeer Alsadoon, N. Costadopoulos, e P. W. C. Prasad. 2017. «Ethical Implications of User Perceptions of Wearable Devices». *Sci. Eng. Ethics* 24 (1): 1–28. https://doi.org/10.1007/s11948-017-9872-8.
Seide, Frank, e Amit Agarwal. 2016. «Cntk: Microsoft's Open-Source Deep-Learning Toolkit». In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2135–35. ACM. https://doi.org/10.1145/2939672.2945397.
Selvaraju, Ramprasaath R., Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, e Dhruv Batra. 2017. «Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization». In *2017 IEEE International Conference on Computer Vision (ICCV)*, 618–26. IEEE. https://doi.org/10.1109/iccv.2017.74.
Seong, Nak Hee, Dong Hyuk Woo, Vijayalakshmi Srinivasan, Jude A. Rivers, e Hsien-Hsin S. Lee. 2010. «SAFER: Stuck-at-fault Error Recovery for Memories». In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, 115–24. IEEE; IEEE. https://doi.org/10.1109/micro.2010.46.
Seyedzadeh, Saleh, Farzad Pour Rahimian, Ivan Glesk, e Marc Roper. 2018. «Machine learning for estimation of building energy consumption and performance: A review». *Visualization in Engineering* 6 (1): 1–20. https://doi.org/10.1186/s40327-018-0064-7.
Shalev-Shwartz, Shai, Shaked Shammah, e Amnon Shashua. 2017. «On a formal model of safe and scalable self-driving cars». *ArXiv preprint* abs/1708.06374. https://arxiv.org/abs/1708.06374.
Shan, Shawn, Wenxin Ding, Josephine Passananti, Haitao Zheng, e Ben Y Zhao. 2023. «Prompt-Specific Poisoning Attacks on Text-to-Image Generative Models». *ArXiv preprint* abs/2310.13828. https://arxiv.org/abs/2310.13828.

Shastri, Bhavin J., Alexander N. Tait, T. Ferreira de Lima, Wolfram H. P. Pernice, Harish Bhaskaran, C. D. Wright, e Paul R. Prucnal. 2021. «Photonics for artificial intelligence and neuromorphic computing». *Nat. Photonics* 15 (2): 102–14. https://doi.org/10.1038/s41566-020-00754-y.
Sheaffer, Jeremy W, David P Luebke, e Kevin Skadron. 2007. «A hardware redundancy and recovery mechanism for reliable scientific computation on graphics processors». In *Graphics Hardware*, 2007:55–64. Citeseer.
Shehabi, Arman, Sarah Smith, Dale Sartor, Richard Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, e William Lintner. 2016. «United states data center energy usage report».
Shen, Sheng, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W. Mahoney, e Kurt Keutzer. 2020. «Q-BERT: Hessian Based Ultra Low Precision Quantization of BERT». *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (05): 8815–21. https://doi.org/10.1609/aaai.v34i05.6409.
Sheng, Victor S., e Jing Zhang. 2019. «Machine Learning with Crowdsourcing: A Brief Summary of the Past Research and Future Directions». *Proceedings of the AAAI Conference on Artificial Intelligence* 33 (01): 9837–43. https://doi.org/10.1609/aaai.v33i01.33019837.
Shi, Hongrui, e Valentin Radu. 2022. «Data selection for efficient model update in federated learning». In *Proceedings of the 2nd European Workshop on Machine Learning and Systems*, 72–78. ACM. https://doi.org/10.1145/3517207.3526980.
Shneiderman, Ben. 2020. «Bridging the Gap Between Ethics and Practice: Guidelines for Reliable, Safe, and Trustworthy Human-centered AI Systems». *ACM Trans. Interact. Intell. Syst.* 10 (4): 1–31. https://doi.org/10.1145/3419764.
———. 2022. *Human-centered AI*. Oxford University Press.
Shokri, Reza, Marco Stronati, Congzheng Song, e Vitaly Shmatikov. 2017. «Membership Inference Attacks Against Machine Learning Models». In *2017 IEEE Symposium on Security and Privacy (SP)*, 3–18. IEEE; IEEE. https://doi.org/10.1109/sp.2017.41.
Siddik, Md Abu Bakar, Arman Shehabi, e Landon Marston. 2021. «The environmental footprint of data centers in the United States». *Environ. Res. Lett.* 16 (6): 064017. https://doi.org/10.1088/1748-9326/abfba1.
Silvestro, Daniele, Stefano Goria, Thomas Sterner, e Alexandre Antonelli. 2022. «Improving biodiversity protection through artificial intelligence». *Nature Sustainability* 5 (5): 415–24. https://doi.org/10.1038/s41893-022-00851-6.
Singh, Narendra, e Oladele A. Ogunseitan. 2022. «Disentangling the worldwide web of e-waste and climate change co-benefits». *Circular Economy* 1 (2): 100011. https://doi.org/10.1016/j.cec.2022.100011.
Skorobogatov, Sergei. 2009. «Local heating attacks on Flash memory devices». In *2009 IEEE International Workshop on Hardware-Oriented Security and Trust*, 1–6. IEEE; IEEE. https://doi.org/10.1109/hst.2009.5225028.
Skorobogatov, Sergei P, e Ross J Anderson. 2003. «Optical fault induction attacks». In *Cryptographic Hardware and Embedded Systems-CHES 2002: 4th International Workshop Redwood Shores, CA, USA, August 1315, 2002 Revised Papers 4*, 2–12. Springer.
Smilkov, Daniel, Nikhil Thorat, Been Kim, Fernanda Viégas, e Martin Wattenberg. 2017. «Smoothgrad: Removing noise by adding noise». *ArXiv preprint* abs/1706.03825. https://arxiv.org/abs/1706.03825.
Snoek, Jasper, Hugo Larochelle, e Ryan P. Adams. 2012. «Practical Bayesian Optimization of Machine Learning Algorithms». In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, a cura di Peter L. Bartlett, Fernando C. N. Pereira, Christopher J. C. Burges, Léon Bottou, e Kilian Q.

Weinberger, 2960–68. https://proceedings.neurips.cc/paper/2012/hash/05311655a15b75fab86956663e1819cd-Abstract.html.
Srivastava, Nitish, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, e Ruslan Salakhutdinov. 2014. «Dropout: a simple way to prevent neural networks from overfitting.» *J. Mach. Learn. Res.* 15 (1): 1929–58. https://doi.org/10.5555/2627435.2670313.
Strubell, Emma, Ananya Ganesh, e Andrew McCallum. 2019. «Energy and Policy Considerations for Deep Learning in NLP». In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 3645–50. Florence, Italy: Association for Computational Linguistics. https://doi.org/10.18653/v1/p19-1355.
Suda, Naveen, Vikas Chandra, Ganesh Dasika, Abinash Mohanty, Yufei Ma, Sarma Vrudhula, Jae-sun Seo, e Yu Cao. 2016. «Throughput-Optimized OpenCL-based FPGA Accelerator for Large-Scale Convolutional Neural Networks». In *Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 16–25. ACM. https://doi.org/10.1145/2847263.2847276.
Sudhakar, Soumya, Vivienne Sze, e Sertac Karaman. 2023. «Data Centers on Wheels: Emissions From Computing Onboard Autonomous Vehicles». *IEEE Micro* 43 (1): 29–39. https://doi.org/10.1109/mm.2022.3219803.
Sze, Vivienne, Yu-Hsin Chen, Tien-Ju Yang, e Joel S. Emer. 2017. «Efficient Processing of Deep Neural Networks: A Tutorial and Survey». *Proc. IEEE* 105 (12): 2295–2329. https://doi.org/10.1109/jproc.2017.2761740.
Szegedy, Christian, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, e Rob Fergus. 2014. «Intriguing properties of neural networks». In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, a cura di Yoshua Bengio e Yann LeCun. http://arxiv.org/abs/1312.6199.
Tambe, Thierry, En-Yu Yang, Zishen Wan, Yuntian Deng, Vijay Janapa Reddi, Alexander Rush, David Brooks, e Gu-Yeon Wei. 2020. «Algorithm-Hardware Co-Design of Adaptive Floating-Point Encodings for Resilient Deep Learning Inference». In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, 1–6. IEEE; IEEE. https://doi.org/10.1109/dac18072.2020.9218516.
Tan, Mingxing, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, e Quoc V. Le. 2019. «MnasNet: Platform-aware Neural Architecture Search for Mobile». In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2820–28. IEEE. https://doi.org/10.1109/cvpr.2019.00293.
Tan, Mingxing, e Quoc V. Le. 2023. «Demystifying Deep Learning». Wiley. https://doi.org/10.1002/9781394205639.ch6.
Tang, Xin, Yichun He, e Jia Liu. 2022. «Soft bioelectronics for cardiac interfaces». *Biophysics Reviews* 3 (1). https://doi.org/10.1063/5.0069516.
Tang, Xin, Hao Shen, Siyuan Zhao, Na Li, e Jia Liu. 2023. «Flexible braincomputer interfaces». *Nature Electronics* 6 (2): 109–18. https://doi.org/10.1038/s41928-022-00913-9.
Tarun, Ayush K, Vikram S Chundawat, Murari Mandal, e Mohan Kankanhalli. 2022. «Deep Regression Unlearning». *ArXiv preprint* abs/2210.08196. https://arxiv.org/abs/2210.08196.
Team, The Theano Development, Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, et al. 2016. «Theano: A Python framework for fast computation of mathematical expressions». https://arxiv.org/abs/1605.02688.
«The Ultimate Guide to Deep Learning Model Quantization and Quantization-Aware Training». s.d. https://deci.ai/quantization-and-quantization-aware-training/.

Thompson, Neil C., Kristjan Greenewald, Keeheon Lee, e Gabriel F. Manso. 2021. «Deep Learning's Diminishing Returns: The Cost of Improvement is Becoming Unsustainable». *IEEE Spectr.* 58 (10): 50–55. https://doi.org/10.1109/mspec.2021.9563954.
Till, Aaron, Andrew L. Rypel, Andrew Bray, e Samuel B. Fey. 2019. «Fish die-offs are concurrent with thermal extremes in north temperate lakes». *Nat. Clim. Change* 9 (8): 637–41. https://doi.org/10.1038/s41558-019-0520-y.
Tirtalistyani, Rose, Murtiningrum Murtiningrum, e Rameshwar S. Kanwar. 2022. «Indonesia Rice Irrigation System: Time for Innovation». *Sustainability* 14 (19): 12477. https://doi.org/10.3390/su141912477.
Tokui, Seiya, Ryosuke Okuta, Takuya Akiba, Yusuke Niitani, Toru Ogawa, Shunta Saito, Shuji Suzuki, Kota Uenishi, Brian Vogel, e Hiroyuki Yamazaki Vincent. 2019. «Chainer: A Deep Learning Framework for Accelerating the Research Cycle». In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 5:1–6. ACM. https://doi.org/10.1145/3292500.3330756.
Tramèr, Florian, Pascal Dupré, Gili Rusak, Giancarlo Pellegrino, e Dan Boneh. 2019. «AdVersarial: Perceptual Ad Blocking meets Adversarial Machine Learning». In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2005–21. ACM. https://doi.org/10.1145/3319535.3354222.
Tran, Cuong, Ferdinando Fioretto, Jung-Eun Kim, e Rakshit Naidu. 2022. «Pruning has a disparate impact on model accuracy». *Adv Neural Inf Process Syst* 35: 17652–64.
Tsai, Min-Jen, Ping-Yi Lin, e Ming-En Lee. 2023. «Adversarial Attacks on Medical Image Classification». *Cancers* 15 (17): 4228. https://doi.org/10.3390/cancers15174228.
Tsai, Timothy, Siva Kumar Sastry Hari, Michael Sullivan, Oreste Villa, e Stephen W. Keckler. 2021. «NVBitFI: Dynamic Fault Injection for GPUs». In *2021 51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 284–91. IEEE; IEEE. https://doi.org/10.1109/dsn48987.2021.00041.
Tschand, Arya, Arun Tejusve Raghunath Rajan, Sachin Idgunji, Anirban Ghosh, Jeremy Holleman, Csaba Kiraly, Pawan Ambalkar, et al. 2024. «MLPerf Power: Benchmarking the Energy Efficiency of Machine Learning Systems from {\mu} Watts to MWatts for Sustainable AI». *arXiv preprint arXiv:2410.12032*.
Uddin, Mueen, e Azizah Abdul Rahman. 2012. «Energy efficiency and low carbon enabler green IT framework for data centers considering green metrics». *Renewable Sustainable Energy Rev.* 16 (6): 4078–94. https://doi.org/10.1016/j.rser.2012.03.014.
Un, e World Economic Forum. 2019. *A New Circular Vision for Electronics, Time for a Global Reboot*. PACE - Platform for Accelerating the Circular Economy. https://www3.weforum.org/docs/WEF_A_New_Circular_Vision_for_Electronics.pdf.
Valenzuela, Christine L, e Pearl Y Wang. 2000. «A genetic algorithm for VLSI floorplanning». In *Parallel Problem Solving from Nature PPSN VI: 6th International Conference Paris, France, September 1820, 2000 Proceedings 6*, 671–80. Springer.
Van Noorden, Richard. 2016. «ArXiv preprint server plans multimillion-dollar overhaul». *Nature* 534 (7609): 602–2. https://doi.org/10.1038/534602a.
Vangal, Sriram, Somnath Paul, Steven Hsu, Amit Agarwal, Saurabh Kumar, Ram Krishnamurthy, Harish Krishnamurthy, James Tschanz, Vivek De, e Chris H. Kim. 2021. «Wide-Range Many-Core SoC Design in Scaled CMOS: Challenges and Opportunities». *IEEE Trans. Very Large Scale Integr. VLSI Syst.* 29 (5): 843–56. https://doi.org/10.1109/tvlsi.2021.3061649.
Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, e Illia Polosukhin. 2017. «Attention is all you need». *Adv Neural Inf Process Syst* 30.
«Vector-borne diseases». s.d. https://www.who.int/news-room/fact-sheets/detail/vector-borne-diseases.

Velazco, Raoul, Gilles Foucard, e Paul Peronnard. 2010. «Combining Results of Accelerated Radiation Tests and Fault Injections to Predict the Error Rate of an Application Implemented in SRAM-Based FPGAs». *IEEE Trans. Nucl. Sci.* 57 (6): 3500–3505. https://doi.org/10.1109/tns.2010.2087355.

Verma, Naveen, Hongyang Jia, Hossein Valavi, Yinqi Tang, Murat Ozatay, Lung-Yen Chen, Bonan Zhang, e Peter Deaville. 2019. «In-Memory Computing: Advances and Prospects». *IEEE Solid-State Circuits Mag.* 11 (3): 43–55. https://doi.org/10.1109/mssc.2019.2922889.

Verma, Team Dual_Boot: Swapnil. 2022. «Elephant AI». *Hackster.io*. https://www.hackster.io/dual_boot/elephant-ai-ba71e9.

Vinuesa, Ricardo, Hossein Azizpour, Iolanda Leite, Madeline Balaam, Virginia Dignum, Sami Domisch, Anna Felländer, Simone Daniela Langhans, Max Tegmark, e Francesco Fuso Nerini. 2020. «The role of artificial intelligence in achieving the Sustainable Development Goals». *Nat. Commun.* 11 (1): 1–10. https://doi.org/10.1038/s41467-019-14108-y.

Vivet, Pascal, Eric Guthmuller, Yvain Thonnart, Gael Pillonnet, Cesar Fuguet, Ivan Miro-Panades, Guillaume Moritz, et al. 2021. «IntAct: A 96-Core Processor With Six Chiplets 3D-Stacked on an Active Interposer With Distributed Interconnects and Integrated Power Management». *IEEE J. Solid-State Circuits* 56 (1): 79–97. https://doi.org/10.1109/jssc.2020.3036341.

Wachter, Sandra, Brent Mittelstadt, e Chris Russell. 2017. «Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR». *SSRN Electronic Journal* 31: 841. https://doi.org/10.2139/ssrn.3063289.

Wald, Peter H., e Jeffrey R. Jones. 1987. «Semiconductor manufacturing: An introduction to processes and hazards». *Am. J. Ind. Med.* 11 (2): 203–21. https://doi.org/10.1002/ajim.4700110209.

Wan, Zishen, Aqeel Anwar, Yu-Shun Hsiao, Tianyu Jia, Vijay Janapa Reddi, e Arijit Raychowdhury. 2021. «Analyzing and Improving Fault Tolerance of Learning-Based Navigation Systems». In *2021 58th ACM/IEEE Design Automation Conference (DAC)*, 841–46. IEEE; IEEE. https://doi.org/10.1109/dac18074.2021.9586116.

Wan, Zishen, Yiming Gan, Bo Yu, S Liu, A Raychowdhury, e Y Zhu. 2023. «Vpp: The vulnerability-proportional protection paradigm towards reliable autonomous machines». In *Proceedings of the 5th International Workshop on Domain Specific System Architecture (DOSSA)*, 1–6.

Wang, LingFeng, e YaQing Zhan. 2019a. «A conceptual peer review model for arXiv and other preprint databases». *Learn. Publ.* 32 (3): 213–19. https://doi.org/10.1002/leap.1229.

———. 2019b. «A conceptual peer review model for arXiv and other preprint databases». *Learn. Publ.* 32 (3): 213–19. https://doi.org/10.1002/leap.1229.

Wang, Tianzhe, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, e Song Han. 2020. «APQ: Joint Search for Network Architecture, Pruning and Quantization Policy». In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2075–84. IEEE. https://doi.org/10.1109/cvpr42600.2020.00215.

Warden, Pete. 2018. «Speech commands: A dataset for limited-vocabulary speech recognition». *arXiv preprint arXiv:1804.03209*.

Warden, Pete, e Daniel Situnayake. 2019. *Tinyml: Machine learning with tensorflow lite on arduino and ultra-low-power microcontrollers*. O'Reilly Media.

Weik, Martin H. 1955. *A Survey of Domestic Electronic Digital Computing Systems*. Ballistic Research Laboratories.

Weiser, Mark. 1991. «The Computer for the 21st Century». *Sci. Am.* 265 (3): 94–104. https://doi.org/10.1038/scientificamerican0991-94.

Wess, Matthias, Matvey Ivanov, Christoph Unger, e Anvesh Nookala. 2020. «ANNETTE: Accurate Neural Network Execution Time Estimation with Stacked Models». *IEEE*. https://doi.org/10.1109/ACCESS.2020.3047259.
Wiener, Norbert. 1960. «Some Moral and Technical Consequences of Automation: As machines learn they may develop unforeseen strategies at rates that baffle their programmers.» *Science* 131 (3410): 1355–58. https://doi.org/10.1126/science.131.3410.1355.
Wilkening, Mark, Vilas Sridharan, Si Li, Fritz Previlon, Sudhanva Gurumurthi, e David R. Kaeli. 2014. «Calculating Architectural Vulnerability Factors for Spatial Multi-Bit Transient Faults». In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, 293–305. IEEE; IEEE. https://doi.org/10.1109/micro.2014.15.
Winkler, Harald, Franck Lecocq, Hans Lofgren, Maria Virginia Vilariño, Sivan Kartha, e Joana Portugal-Pereira. 2022. «Examples of shifting development pathways: Lessons on how to enable broader, deeper, and faster climate action». *Climate Action* 1 (1). https://doi.org/10.1007/s44168-022-00026-1.
Wong, H.-S. Philip, Heng-Yuan Lee, Shimeng Yu, Yu-Sheng Chen, Yi Wu, Pang-Shiu Chen, Byoungil Lee, Frederick T. Chen, e Ming-Jinn Tsai. 2012. «MetalOxide RRAM». *Proc. IEEE* 100 (6): 1951–70. https://doi.org/10.1109/jproc.2012.2190369.
Wu, Bichen, Kurt Keutzer, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, e Yangqing Jia. 2019. «FBNet: Hardware-aware Efficient ConvNet Design via Differentiable Neural Architecture Search». In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10734–42. IEEE. https://doi.org/10.1109/cvpr.2019.01099.
Wu, Carole-Jean, David Brooks, Kevin Chen, Douglas Chen, Sy Choudhury, Marat Dukhan, Kim Hazelwood, et al. 2019. «Machine Learning at Facebook: Understanding Inference at the Edge». In *2019 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, 331–44. IEEE; IEEE. https://doi.org/10.1109/hpca.2019.00048.
Wu, Carole-Jean, Ramya Raghavendra, Udit Gupta, Bilge Acun, Newsha Ardalani, Kiwan Maeng, Gloria Chang, et al. 2022. «Sustainable ai: Environmental implications, challenges and opportunities». *Proceedings of Machine Learning and Systems* 4: 795–813.
Wu, Zhang Judd, e Micikevicius Isaev. 2020. «Integer Quantization for Deep Learning Inference: Principles and Empirical Evaluation)». *ArXiv preprint*. https://arxiv.org/abs/2004.09602.
Xiao, Seznec Lin, Demouth Wu, e Han. 2022. «SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models». *ArXiv preprint*. https://arxiv.org/abs/2211.10438.
Xie, Cihang, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, e Quoc V. Le. 2020. «Adversarial Examples Improve Image Recognition». In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 816–25. IEEE. https://doi.org/10.1109/cvpr42600.2020.00090.
Xie, Saining, Ross Girshick, Piotr Dollar, Zhuowen Tu, e Kaiming He. 2017. «Aggregated Residual Transformations for Deep Neural Networks». In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1492–1500. IEEE. https://doi.org/10.1109/cvpr.2017.634.
Xinyu, Chen. s.d.
Xiong, Siyu, Guoqing Wu, Xitian Fan, Xuan Feng, Zhongcheng Huang, Wei Cao, Xuegong Zhou, et al. 2021. «MRI-based brain tumor segmentation using FPGA-accelerated neural network». *BMC Bioinf.* 22 (1): 421. https://doi.org/10.1186/s12859-021-04347-6.
Xiu, Liming. 2019. «Time Moore: Exploiting Moore's Law From The Perspective of Time». *IEEE Solid-State Circuits Mag.* 11 (1): 39–55. https://doi.org/10.1109/mssc.2018.2882285.

Xu, Chen, Jianqiang Yao, Zhouchen Lin, Wenwu Ou, Yuanbin Cao, Zhirong Wang, e Hongbin Zha. 2018. «Alternating Multi-bit Quantization for Recurrent Neural Networks». In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. https://openreview.net/forum?id=S19dR9x0b.

Xu, Hu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, e Christoph Feichtenhofer. 2023. «Demystifying CLIP Data». *ArXiv preprint* abs/2309.16671. https://arxiv.org/abs/2309.16671.

Xu, Ying, Xu Zhong, Antonio Jimeno Yepes, e Jey Han Lau. 2021. «Grey-box adversarial attack and defence for sentiment classification». *arXiv preprint arXiv:2103.11576*.

Xu, Zheng, Yanxiang Zhang, Galen Andrew, Christopher A. Choquette-Choo, Peter Kairouz, H. Brendan McMahan, Jesse Rosenstock, e Yuanbo Zhang. 2023. «Federated Learning of Gboard Language Models with Differential Privacy». *ArXiv preprint* abs/2305.18465 (maggio). http://arxiv.org/abs/2305.18465v2.

Yang, Tien-Ju, Yonghui Xiao, Giovanni Motta, Françoise Beaufays, Rajiv Mathews, e Mingqing Chen. 2023. «Online Model Compression for Federated Learning with Large Models». In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1–5. IEEE; IEEE. https://doi.org/10.1109/icassp49357.2023.10097124.

Yao, Zhewei, Zhen Dong, Zhangcheng Zheng, Amir Gholami, Jiali Yu, Eric Tan, Leyuan Wang, et al. 2021. «Hawq-v3: Dyadic neural network quantization». In *International Conference on Machine Learning*, 11875–86. PMLR.

Ye, Linfeng, e Shayan Mohajer Hamidi. 2021. «Thundernna: A white box adversarial attack». *arXiv preprint arXiv:2111.12305*.

Yeh, Y. C. 1996. «Triple-triple redundant 777 primary flight computer». In *1996 IEEE Aerospace Applications Conference. Proceedings*, 1:293–307. IEEE; IEEE. https://doi.org/10.1109/aero.1996.495891.

Yik, Jason, Soikat Hasan Ahmed, Zergham Ahmed, Brian Anderson, Andreas G. Andreou, Chiara Bartolozzi, Arindam Basu, et al. 2023. «NeuroBench: Advancing Neuromorphic Computing through Collaborative, Fair and Representative Benchmarking». https://arxiv.org/abs/2304.04640.

You, Jie, Jae-Won Chung, e Mosharaf Chowdhury. 2023. «Zeus: Understanding and Optimizing GPU Energy Consumption of DNN Training». In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 119–39. Boston, MA: USENIX Association. https://www.usenix.org/conference/nsdi23/presentation/you.

You, Yang, Zhao Zhang, Cho-Jui Hsieh, James Demmel, e Kurt Keutzer. 2017. «ImageNet Training in Minutes», settembre. http://arxiv.org/abs/1709.05011v10.

Young, Tom, Devamanyu Hazarika, Soujanya Poria, e Erik Cambria. 2018. «Recent Trends in Deep Learning Based Natural Language Processing [Review Article]». *IEEE Comput. Intell. Mag.* 13 (3): 55–75. https://doi.org/10.1109/mci.2018.2840738.

Yu, Yuan, Martín Abadi, Paul Barham, Eugene Brevdo, Mike Burrows, Andy Davis, Jeff Dean, et al. 2018. «Dynamic control flow in large-scale machine learning». In *Proceedings of the Thirteenth EuroSys Conference*, 265–83. ACM. https://doi.org/10.1145/3190508.3190551.

Zafrir, Ofir, Guy Boudoukh, Peter Izsak, e Moshe Wasserblat. 2019. «Q8BERT: Quantized 8Bit BERT». In *2019 Fifth Workshop on Energy Efficient Machine Learning and Cognitive Computing - NeurIPS Edition (EMC2-NIPS)*, 36–39. IEEE; IEEE. https://doi.org/10.1109/emc2-nips53020.2019.00016.

Zeiler, Matthew D. 2012. «ADADELTA: An Adaptive Learning Rate Method», dicembre, 119–49. https://doi.org/10.1002/9781118266502.ch6.

Zennaro, Marco, Brian Plancher, e V Janapa Reddi. 2022. «TinyML: Applied AI for development». In *The UN 7th Multi-stakeholder Forum on Science, Technology and Innovation for the Sustainable Development Goals*, 2022–05.
Zhang, Chengliang, Minchen Yu, Wei Wang 0030, e Feng Yan 0001. 2019. «MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving.» In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, 1049–62. https://www.usenix.org/conference/atc19/presentation/zhang-chengliang.
Zhang, Chen, Peng Li, Guangyu Sun, Yijin Guan, Bingjun Xiao, e Jason Optimizing Cong. 2015. «FPGA-based Accelerator Design for Deep Convolutional Neural Networks Proceedings of the 2015 ACM». In *SIGDA International Symposium on Field-Programmable Gate Arrays-FPGA*, 15:161–70.
Zhang, Dan, Safeen Huda, Ebrahim Songhori, Kartik Prabhu, Quoc Le, Anna Goldie, e Azalia Mirhoseini. 2022. «A full-stack search technique for domain optimized deep learning accelerators». In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, 27–42. ASPLOS '22. New York, NY, USA: ACM. https://doi.org/10.1145/3503222.3507767.
Zhang, Dongxia, Xiaoqing Han, e Chunyu Deng. 2018. «Review on the research and practice of deep learning and reinforcement learning in smart grids». *CSEE Journal of Power and Energy Systems* 4 (3): 362–70. https://doi.org/10.17775/cseejpes.2018.00520.
Zhang, Hongyu. 2008. «On the Distribution of Software Faults». *IEEE Trans. Software Eng.* 34 (2): 301–2. https://doi.org/10.1109/tse.2007.70771.
Zhang, Jeff Jun, Tianyu Gu, Kanad Basu, e Siddharth Garg. 2018. «Analyzing and mitigating the impact of permanent faults on a systolic array based neural network accelerator». In *2018 IEEE 36th VLSI Test Symposium (VTS)*, 1–6. IEEE; IEEE. https://doi.org/10.1109/vts.2018.8368656.
Zhang, Jeff, Kartheek Rangineni, Zahra Ghodsi, e Siddharth Garg. 2018. «ThUnderVolt: Enabling Aggressive Voltage Underscaling and Timing Error Resilience for Energy Efficient Deep Learning Accelerators». In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*, 1–6. IEEE. https://doi.org/10.1109/dac.2018.8465918.
Zhang, Li Lyna, Yuqing Yang, Yuhang Jiang, Wenwu Zhu, e Yunxin Liu. 2020. «Fast Hardware-Aware Neural Architecture Search». In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE. https://doi.org/10.1109/cvprw50498.2020.00354.
Zhang, Qingxue, Dian Zhou, e Xuan Zeng. 2017. «Highly wearable cuff-less blood pressure and heart rate monitoring with single-arm electrocardiogram and photoplethysmogram signals». *BioMedical Engineering OnLine* 16 (1): 23. https://doi.org/10.1186/s12938-017-0317-z.
Zhang, Tunhou, Hsin-Pai Cheng, Zhenwen Li, Feng Yan, Chengyu Huang, Hai Helen Li, e Yiran Chen. 2020. «AutoShrink: A Topology-Aware NAS for Discovering Efficient Neural Architecture». In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 6829–36. AAAI Press. https://aaai.org/ojs/index.php/AAAI/article/view/6163.
Zhao, Mark, e G. Edward Suh. 2018. «FPGA-Based Remote Power Side-Channel Attacks». In *2018 IEEE Symposium on Security and Privacy (SP)*, 229–44. IEEE; IEEE. https://doi.org/10.1109/sp.2018.00049.
Zhao, Yue, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, e Vikas Chandra. 2018. «Federated Learning with Non-IID Data». *ArXiv preprint* abs/1806.00582 (giugno). http://arxiv.org/abs/1806.00582v2.

Zhou, Bolei, Yiyou Sun, David Bau, e Antonio Torralba. 2018. «Interpretable basis decomposition for visual explanation». In *Proceedings of the European Conference on Computer Vision (ECCV)*, 119–34.
Zhou, Chuteng, Fernando Garcia Redondo, Julian Büchel, Irem Boybat, Xavier Timoneda Comas, S. R. Nandakumar, Shidhartha Das, Abu Sebastian, Manuel Le Gallo, e Paul N. Whatmough. 2021. «AnalogNets: Ml-hw Co-Design of Noise-robust TinyML Models and Always-On Analog Compute-in-Memory Accelerator». https://arxiv.org/abs/2111.06503.
Zhou, Guanglei, e Jason H. Anderson. 2023. «Area-Driven FPGA Logic Synthesis Using Reinforcement Learning». In *Proceedings of the 28th Asia and South Pacific Design Automation Conference*, 159–65. ACM. https://doi.org/10.1145/3566097.3567894.
Zhou, Hailong, Jianji Dong, Junwei Cheng, Wenchan Dong, Chaoran Huang, Yichen Shen, Qiming Zhang, et al. 2022. «Photonic matrix multiplication lights up photonic accelerator and beyond». *Light: Science & Applications* 11 (1): 30. https://doi.org/10.1038/s41377-022-00717-8.
Zhou, Peng, Xintong Han, Vlad I. Morariu, e Larry S. Davis. 2018. «Learning Rich Features for Image Manipulation Detection». In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1053–61. IEEE. https://doi.org/10.1109/cvpr.2018.00116.
Zhu, Hongyu, Mohamed Akrout, Bojian Zheng, Andrew Pelegris, Anand Jayarajan, Amar Phanishayee, Bianca Schroeder, e Gennady Pekhimenko. 2018. «Benchmarking and Analyzing Deep Neural Network Training». In *2018 IEEE International Symposium on Workload Characterization (IISWC)*, 88–100. IEEE; IEEE. https://doi.org/10.1109/iiswc.2018.8573476.
Zhu, Ligeng, Lanxiang Hu, Ji Lin, Wei-Ming Chen, Wei-Chen Wang, Chuang Gan, e Song Han. 2023. «PockEngine: Sparse and Efficient Fine-tuning in a Pocket». In *56th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM. https://doi.org/10.1145/3613424.3614307.
Zhuang, Fuzhen, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, e Qing He. 2021. «A Comprehensive Survey on Transfer Learning». *Proceedings of the IEEE* 109 (1): 43–76. https://doi.org/10.1109/jproc.2020.3004555.
Zoph, Barret, e Quoc V. Le. 2016. «Neural Architecture Search with Reinforcement Learning», novembre, 367–92. https://doi.org/10.1002/9781394217519.ch17.