Dive into Deep Learning

Release 0.16.1

Aston Zhang, Zachary C. Lipton, Mu Li, and Alexander J. Smola

Jan 19, 2021

Contents

Preface **1**

Installation **9**

Notation **13**

1 Introduction **17**

1.1 A Motivating Example . . . 18
1.2 Key Components . . . 20
1.3 Kinds of Machine Learning Problems . . . 22
1.4 Roots . . . 34
1.5 The Road to Deep Learning . . . 36
1.6 Success Stories . . . 38
1.7 Characteristics . . . 40

2 Preliminaries **43**

2.1 Data Manipulation . . . 43
2.1.1 Getting Started . . . 44
2.1.2 Operations . . . 46
2.1.3 Broadcasting Mechanism . . . 48
2.1.4 Indexing and Slicing . . . 48
2.1.5 Saving Memory . . . 49
2.1.6 Conversion to Other Python Objects . . . 50
2.2 Data Preprocessing . . . 51
2.2.1 Reading the Dataset . . . 51
2.2.2 Handling Missing Data . . . 52
2.2.3 Conversion to the Tensor Format . . . 53
2.3 Linear Algebra . . . 53
2.3.1 Scalars . . . 54
2.3.2 Vectors . . . 54
2.3.3 Matrices . . . 56
2.3.4 Tensors . . . 57
2.3.5 Basic Properties of Tensor Arithmetic . . . 58
2.3.6 Reduction . . . 59
2.3.7 Dot Products . . . 61
2.3.8 Matrix-Vector Products . . . 61
2.3.9 Matrix-Matrix Multiplication . . . 62
2.3.10 Norms . . . 63
2.3.11 More on Linear Algebra . . . 65
2.4 Calculus . . . 66
2.4.1 Derivatives and Differentiation . . . 67

2.4.2 Partial Derivatives . . . 70
2.4.3 Gradients . . . 70
2.4.4 Chain Rule . . . 71
2.5 Automatic Differentiation . . . 72
2.5.1 A Simple Example . . . 72
2.5.2 Backward for Non-Scalar Variables . . . 73
2.5.3 Detaching Computation . . . 74
2.5.4 Computing the Gradient of Python Control Flow . . . 74
2.6 Probability . . . 76
2.6.1 Basic Probability Theory . . . 77
2.6.2 Dealing with Multiple Random Variables . . . 80
2.6.3 Expectation and Variance . . . 83
2.7 Documentation . . . 84
2.7.1 Finding All the Functions and Classes in a Module . . . 84
2.7.2 Finding the Usage of Specific Functions and Classes . . . 85

3 Linear Neural Networks **87**
3.1 Linear Regression . . . 87
3.1.1 Basic Elements of Linear Regression . . . 87
3.1.2 Vectorization for Speed . . . 91
3.1.3 The Normal Distribution and Squared Loss . . . 93
3.1.4 From Linear Regression to Deep Networks . . . 94
3.2 Linear Regression Implementation from Scratch . . . 97
3.2.1 Generating the Dataset . . . 97
3.2.2 Reading the Dataset . . . 98
3.2.3 Initializing Model Parameters . . . 99
3.2.4 Defining the Model . . . 100
3.2.5 Defining the Loss Function . . . 100
3.2.6 Defining the Optimization Algorithm . . . 100
3.2.7 Training . . . 101
3.3 Concise Implementation of Linear Regression . . . 103
3.3.1 Generating the Dataset . . . 103
3.3.2 Reading the Dataset . . . 103
3.3.3 Defining the Model . . . 104
3.3.4 Initializing Model Parameters . . . 105
3.3.5 Defining the Loss Function . . . 105
3.3.6 Defining the Optimization Algorithm . . . 105
3.3.7 Training . . . 106
3.4 Softmax Regression . . . 107
3.4.1 Classification Problem . . . 108
3.4.2 Network Architecture . . . 108
3.4.3 Parameterization Cost of Fully-Connected Layers . . . 109
3.4.4 Softmax Operation . . . 109
3.4.5 Vectorization for Minibatches . . . 110
3.4.6 Loss Function . . . 110
3.4.7 Information Theory Basics . . . 112
3.4.8 Model Prediction and Evaluation . . . 113
3.5 The Image Classification Dataset . . . 114
3.5.1 Reading the Dataset . . . 114
3.5.2 Reading a Minibatch . . . 115
3.5.3 Putting All Things Together . . . 116

3.6 Implementation of Softmax Regression from Scratch . . . 117
3.6.1 Initializing Model Parameters . . . 117
3.6.2 Defining the Softmax Operation . . . 118
3.6.3 Defining the Model . . . 119
3.6.4 Defining the Loss Function . . . 119
3.6.5 Classification Accuracy . . . 120
3.6.6 Training . . . 121
3.6.7 Prediction . . . 123
3.7 Concise Implementation of Softmax Regression . . . 124
3.7.1 Initializing Model Parameters . . . 125
3.7.2 Softmax Implementation Revisited . . . 125
3.7.3 Optimization Algorithm . . . 126
3.7.4 Training . . . 126

4 Multilayer Perceptrons **129**
4.1 Multilayer Perceptrons . . . 129
4.1.1 Hidden Layers . . . 129
4.1.2 Activation Functions . . . 132
4.2 Implementation of Multilayer Perceptrons from Scratch . . . 138
4.2.1 Initializing Model Parameters . . . 138
4.2.2 Activation Function . . . 138
4.2.3 Model . . . 139
4.2.4 Loss Function . . . 139
4.2.5 Training . . . 139
4.3 Concise Implementation of Multilayer Perceptrons . . . 140
4.3.1 Model . . . 141
4.4 Model Selection, Underfitting, and Overfitting . . . 142
4.4.1 Training Error and Generalization Error . . . 143
4.4.2 Model Selection . . . 145
4.4.3 Underfitting or Overfitting? . . . 146
4.4.4 Polynomial Regression . . . 148
4.5 Weight Decay . . . 152
4.5.1 Norms and Weight Decay . . . 153
4.5.2 High-Dimensional Linear Regression . . . 154
4.5.3 Implementation from Scratch . . . 155
4.5.4 Concise Implementation . . . 157
4.6 Dropout . . . 159
4.6.1 Overfitting Revisited . . . 160
4.6.2 Robustness through Perturbations . . . 160
4.6.3 Dropout in Practice . . . 161
4.6.4 Implementation from Scratch . . . 162
4.6.5 Concise Implementation . . . 164
4.7 Forward Propagation, Backward Propagation, and Computational Graphs . . . 166
4.7.1 Forward Propagation . . . 166
4.7.2 Computational Graph of Forward Propagation . . . 167
4.7.3 Backpropagation . . . 167
4.7.4 Training Neural Networks . . . 168
4.8 Numerical Stability and Initialization . . . 170
4.8.1 Vanishing and Exploding Gradients . . . 170
4.8.2 Parameter Initialization . . . 173
4.9 Environment and Distribution Shift . . . 175

4.9.1 Types of Distribution Shift 176
4.9.2 Examples of Distribution Shift 178
4.9.3 Correction of Distribution Shift 180
4.9.4 A Taxonomy of Learning Problems 183
4.9.5 Fairness, Accountability, and Transparency in Machine Learning 185
4.10 Predicting House Prices on Kaggle 186
4.10.1 Downloading and Caching Datasets 186
4.10.2 Kaggle 188
4.10.3 Accessing and Reading the Dataset 189
4.10.4 Data Preprocessing 190
4.10.5 Training 191
4.10.6 K-Fold Cross-Validation 192
4.10.7 Model Selection 193
4.10.8 Submitting Predictions on Kaggle 194

5 Deep Learning Computation **197**
5.1 Layers and Blocks 197
5.1.1 A Custom Block 199
5.1.2 The Sequential Block 201
5.1.3 Executing Code in the Forward Propagation Function 202
5.1.4 Efficiency 203
5.2 Parameter Management 204
5.2.1 Parameter Access 205
5.2.2 Parameter Initialization 208
5.2.3 Tied Parameters 210
5.3 Deferred Initialization 211
5.3.1 Instantiating a Network 212
5.4 Custom Layers 214
5.4.1 Layers without Parameters 214
5.4.2 Layers with Parameters 215
5.5 File I/O 216
5.5.1 Loading and Saving Tensors 216
5.5.2 Loading and Saving Model Parameters 217
5.6 GPUs 219
5.6.1 Computing Devices 220
5.6.2 Tensors and GPUs 221
5.6.3 Neural Networks and GPUs 223

6 Convolutional Neural Networks **225**
6.1 From Fully-Connected Layers to Convolutions 226
6.1.1 Invariance 226
6.1.2 Constraining the MLP 227
6.1.3 Convolutions 229
6.1.4 “Where’s Waldo” Revisited 229
6.2 Convolutions for Images 231
6.2.1 The Cross-Correlation Operation 231
6.2.2 Convolutional Layers 233
6.2.3 Object Edge Detection in Images 233
6.2.4 Learning a Kernel 234
6.2.5 Cross-Correlation and Convolution 235
6.2.6 Feature Map and Receptive Field 236

6.3 Padding and Stride . . . 237
6.3.1 Padding . . . 237
6.3.2 Stride . . . 239
6.4 Multiple Input and Multiple Output Channels . . . 241
6.4.1 Multiple Input Channels . . . 241
6.4.2 Multiple Output Channels . . . 242
6.4.3 1×1 Convolutional Layer . . . 243
6.5 Pooling . . . 245
6.5.1 Maximum Pooling and Average Pooling . . . 245
6.5.2 Padding and Stride . . . 247
6.5.3 Multiple Channels . . . 248
6.6 Convolutional Neural Networks (LeNet) . . . 249
6.6.1 LeNet . . . 250
6.6.2 Training . . . 252

7 Modern Convolutional Neural Networks 255
7.1 Deep Convolutional Neural Networks (AlexNet) . . . 255
7.1.1 Learning Representations . . . 256
7.1.2 AlexNet . . . 259
7.1.3 Reading the Dataset . . . 262
7.1.4 Training . . . 262
7.2 Networks Using Blocks (VGG) . . . 263
7.2.1 VGG Blocks . . . 264
7.2.2 VGG Network . . . 264
7.2.3 Training . . . 266
7.3 Network in Network (NiN) . . . 268
7.3.1 NiN Blocks . . . 268
7.3.2 NiN Model . . . 270
7.3.3 Training . . . 271
7.4 Networks with Parallel Concatenations (GoogLeNet) . . . 272
7.4.1 Inception Blocks . . . 272
7.4.2 GoogLeNet Model . . . 273
7.4.3 Training . . . 276
7.5 Batch Normalization . . . 277
7.5.1 Training Deep Networks . . . 277
7.5.2 Batch Normalization Layers . . . 279
7.5.3 Implementation from Scratch . . . 280
7.5.4 Applying Batch Normalization in LeNet . . . 281
7.5.5 Concise Implementation . . . 283
7.5.6 Controversy . . . 284
7.6 Residual Networks (ResNet) . . . 285
7.6.1 Function Classes . . . 285
7.6.2 Residual Blocks . . . 286
7.6.3 ResNet Model . . . 289
7.6.4 Training . . . 291
7.7 Densely Connected Networks (DenseNet) . . . 292
7.7.1 From ResNet to DenseNet . . . 292
7.7.2 Dense Blocks . . . 293
7.7.3 Transition Layers . . . 294
7.7.4 DenseNet Model . . . 295
7.7.5 Training . . . 296

8 Recurrent Neural Networks 299
8.1 Sequence Models . . . 299
8.1.1 Statistical Tools . . . 301
8.1.2 Training . . . 303
8.1.3 Prediction . . . 305
8.2 Text Preprocessing . . . 308
8.2.1 Reading the Dataset . . . 309
8.2.2 Tokenization . . . 309
8.2.3 Vocabulary . . . 310
8.2.4 Putting All Things Together . . . 311
8.3 Language Models and the Dataset . . . 312
8.3.1 Learning a Language Model . . . 313
8.3.2 Markov Models and n-grams . . . 314
8.3.3 Natural Language Statistics . . . 314
8.3.4 Reading Long Sequence Data . . . 317
8.4 Recurrent Neural Networks . . . 321
8.4.1 Neural Networks without Hidden States . . . 322
8.4.2 Recurrent Neural Networks with Hidden States . . . 322
8.4.3 RNN-based Character-Level Language Models . . . 324
8.4.4 Perplexity . . . 325
8.5 Implementation of Recurrent Neural Networks from Scratch . . . 327
8.5.1 One-Hot Encoding . . . 327
8.5.2 Initializing the Model Parameters . . . 328
8.5.3 RNN Model . . . 328
8.5.4 Prediction . . . 330
8.5.5 Gradient Clipping . . . 330
8.5.6 Training . . . 331
8.6 Concise Implementation of Recurrent Neural Networks . . . 335
8.6.1 Defining the Model . . . 335
8.6.2 Training and Predicting . . . 337
8.7 Backpropagation Through Time . . . 338
8.7.1 Analysis of Gradients in RNNs . . . 338
8.7.2 Backpropagation Through Time in Detail . . . 341

9 Modern Recurrent Neural Networks 345
9.1 Gated Recurrent Units (GRU) . . . 345
9.1.1 Gated Hidden State . . . 346
9.1.2 Implementation from Scratch . . . 349
9.1.3 Concise Implementation . . . 351
9.2 Long Short-Term Memory (LSTM) . . . 352
9.2.1 Gated Memory Cell . . . 353
9.2.2 Implementation from Scratch . . . 356
9.2.3 Concise Implementation . . . 358
9.3 Deep Recurrent Neural Networks . . . 359
9.3.1 Functional Dependencies . . . 360
9.3.2 Concise Implementation . . . 361
9.3.3 Training and Prediction . . . 361
9.4 Bidirectional Recurrent Neural Networks . . . 363
9.4.1 Dynamic Programming in Hidden Markov Models . . . 363
9.4.2 Bidirectional Model . . . 365
9.4.3 Training a Bidirectional RNN for a Wrong Application . . . 367

9.5 Machine Translation and the Dataset . . . 368
9.5.1 Downloading and Preprocessing the Dataset . . . 369
9.5.2 Tokenization . . . 370
9.5.3 Vocabulary . . . 371
9.5.4 Loading the Dataset . . . 372
9.5.5 Putting All Things Together . . . 373
9.6 Encoder-Decoder Architecture . . . 374
9.6.1 Encoder . . . 374
9.6.2 Decoder . . . 375
9.6.3 Putting the Encoder and Decoder Together . . . 375
9.7 Sequence to Sequence Learning . . . 376
9.7.1 Encoder . . . 377
9.7.2 Decoder . . . 379
9.7.3 Loss Function . . . 380
9.7.4 Training . . . 381
9.7.5 Prediction . . . 383
9.7.6 Evaluation of Predicted Sequences . . . 384
9.8 Beam Search . . . 386
9.8.1 Greedy Search . . . 386
9.8.2 Exhaustive Search . . . 387
9.8.3 Beam Search . . . 388

10 Attention Mechanisms 391
10.1 Attention Cues . . . 391
10.1.1 Attention Cues in Biology . . . 392
10.1.2 Queries, Keys, and Values . . . 393
10.1.3 Visualization of Attention . . . 394
10.2 Attention Pooling: Nadaraya-Watson Kernel Regression . . . 396
10.2.1 Generating the Dataset . . . 396
10.2.2 Average Pooling . . . 397
10.2.3 Nonparametric Attention Pooling . . . 398
10.2.4 Parametric Attention Pooling . . . 400
10.3 Attention Scoring Functions . . . 403
10.3.1 Masked Softmax Operation . . . 405
10.3.2 Additive Attention . . . 406
10.3.3 Scaled Dot-Product Attention . . . 407
10.4 Bahdanau Attention . . . 409
10.4.1 Model . . . 410
10.4.2 Defining the Decoder with Attention . . . 410
10.4.3 Training . . . 412
10.5 Multi-Head Attention . . . 414
10.5.1 Model . . . 415
10.5.2 Implementation . . . 415
10.6 Self-Attention and Positional Encoding . . . 418
10.6.1 Self-Attention . . . 418
10.6.2 Comparing CNNs, RNNs, and Self-Attention . . . 418
10.6.3 Positional Encoding . . . 420
10.7 Transformer . . . 423
10.7.1 Model . . . 423
10.7.2 Positionwise Feed-Forward Networks . . . 425
10.7.3 Residual Connection and Layer Normalization . . . 426

10.7.4 Encoder . . . 427
10.7.5 Decoder . . . 428
10.7.6 Training . . . 430

11 Optimization Algorithms **435**
11.1 Optimization and Deep Learning . . . 435
11.1.1 Optimization and Estimation . . . 436
11.1.2 Optimization Challenges in Deep Learning . . . 437
11.2 Convexity . . . 441
11.2.1 Basics . . . 441
11.2.2 Properties . . . 444
11.2.3 Constraints . . . 447
11.3 Gradient Descent . . . 450
11.3.1 Gradient Descent in One Dimension . . . 450
11.3.2 Multivariate Gradient Descent . . . 453
11.3.3 Adaptive Methods . . . 455
11.4 Stochastic Gradient Descent . . . 459
11.4.1 Stochastic Gradient Updates . . . 459
11.4.2 Dynamic Learning Rate . . . 461
11.4.3 Convergence Analysis for Convex Objectives . . . 462
11.4.4 Stochastic Gradients and Finite Samples . . . 464
11.5 Minibatch Stochastic Gradient Descent . . . 465
11.5.1 Vectorization and Caches . . . 465
11.5.2 Minibatches . . . 468
11.5.3 Reading the Dataset . . . 469
11.5.4 Implementation from Scratch . . . 469
11.5.5 Concise Implementation . . . 473
11.6 Momentum . . . 474
11.6.1 Basics . . . 475
11.6.2 Practical Experiments . . . 479
11.6.3 Theoretical Analysis . . . 481
11.7 Adagrad . . . 484
11.7.1 Sparse Features and Learning Rates . . . 484
11.7.2 Preconditioning . . . 485
11.7.3 The Algorithm . . . 486
11.7.4 Implementation from Scratch . . . 488
11.7.5 Concise Implementation . . . 489
11.8 RMSProp . . . 490
11.8.1 The Algorithm . . . 491
11.8.2 Implementation from Scratch . . . 492
11.8.3 Concise Implementation . . . 493
11.9 Adadelta . . . 494
11.9.1 The Algorithm . . . 494
11.9.2 Implementation . . . 495
11.10 Adam . . . 497
11.10.1 The Algorithm . . . 497
11.10.2 Implementation . . . 498
11.10.3 Yogi . . . 500
11.11 Learning Rate Scheduling . . . 501
11.11.1 Toy Problem . . . 502
11.11.2 Schedulers . . . 503

11.11.3 Policies 505

12 Computational Performance **511**
12.1 Compilers and Interpreters 511
12.1.1 Symbolic Programming 512
12.1.2 Hybrid Programming 513
12.1.3 HybridSequential 514
12.2 Asynchronous Computation 518
12.2.1 Asynchrony via Backend 519
12.2.2 Barriers and Blockers 521
12.2.3 Improving Computation 522
12.2.4 Improving Memory Footprint 522
12.3 Automatic Parallelism 525
12.3.1 Parallel Computation on GPUs 526
12.3.2 Parallel Computation and Communication 527
12.4 Hardware 529
12.4.1 Computers 530
12.4.2 Memory 531
12.4.3 Storage 532
12.4.4 CPUs 533
12.4.5 GPUs and other Accelerators 536
12.4.6 Networks and Buses 538
12.4.7 More Latency Numbers 540
12.5 Training on Multiple GPUs 542
12.5.1 Splitting the Problem 542
12.5.2 Data Parallelism 544
12.5.3 A Toy Network 545
12.5.4 Data Synchronization 546
12.5.5 Distributing Data 547
12.5.6 Training 548
12.5.7 Experiment 549
12.6 Concise Implementation for Multiple GPUs 550
12.6.1 A Toy Network 551
12.6.2 Parameter Initialization and Logistics 551
12.6.3 Training 553
12.6.4 Experiments 554
12.7 Parameter Servers 555
12.7.1 Data Parallel Training 556
12.7.2 Ring Synchronization 558
12.7.3 Multi-Machine Training 561
12.7.4 (key,value) Stores 563

13 Computer Vision **565**
13.1 Image Augmentation 565
13.1.1 Common Image Augmentation Method 566
13.1.2 Using an Image Augmentation Training Model 570
13.2 Fine-Tuning 574
13.2.1 Hot Dog Recognition 575
13.3 Object Detection and Bounding Boxes 580
13.3.1 Bounding Box 581
13.4 Anchor Boxes 583

13.4.1 Generating Multiple Anchor Boxes . . . 583
13.4.2 Intersection over Union . . . 586
13.4.3 Labeling Training Set Anchor Boxes . . . 587
13.4.4 Bounding Boxes for Prediction . . . 592
13.5 Multiscale Object Detection . . . 596
13.6 The Object Detection Dataset . . . 599
13.6.1 Downloading the Dataset . . . 599
13.6.2 Reading the Dataset . . . 600
13.6.3 Demonstration . . . 601
13.7 Single Shot Multibox Detection (SSD) . . . 602
13.7.1 Model . . . 602
13.7.2 Training . . . 608
13.7.3 Prediction . . . 610
13.8 Region-based CNNs (R-CNNs) . . . 613
13.8.1 R-CNNs . . . 613
13.8.2 Fast R-CNN . . . 614
13.8.3 Faster R-CNN . . . 617
13.8.4 Mask R-CNN . . . 618
13.9 Semantic Segmentation and the Dataset . . . 619
13.9.1 Image Segmentation and Instance Segmentation . . . 619
13.9.2 The Pascal VOC2012 Semantic Segmentation Dataset . . . 620
13.10 Transposed Convolution . . . 625
13.10.1 Basic 2D Transposed Convolution . . . 625
13.10.2 Padding, Strides, and Channels . . . 626
13.10.3 Analogy to Matrix Transposition . . . 627
13.11 Fully Convolutional Networks (FCN) . . . 628
13.11.1 Constructing a Model . . . 629
13.11.2 Initializing the Transposed Convolution Layer . . . 631
13.11.3 Reading the Dataset . . . 632
13.11.4 Training . . . 632
13.11.5 Prediction . . . 633
13.12 Neural Style Transfer . . . 635
13.12.1 Technique . . . 636
13.12.2 Reading the Content and Style Images . . . 637
13.12.3 Preprocessing and Postprocessing . . . 638
13.12.4 Extracting Features . . . 638
13.12.5 Defining the Loss Function . . . 639
13.12.6 Creating and Initializing the Composite Image . . . 641
13.12.7 Training . . . 642
13.13 Image Classification (CIFAR-10) on Kaggle . . . 645
13.13.1 Obtaining and Organizing the Dataset . . . 646
13.13.2 Image Augmentation . . . 648
13.13.3 Reading the Dataset . . . 649
13.13.4 Defining the Model . . . 650
13.13.5 Defining the Training Functions . . . 651
13.13.6 Training and Validating the Model . . . 652
13.13.7 Classifying the Testing Set and Submitting Results on Kaggle . . . 652
13.14 Dog Breed Identification (ImageNet Dogs) on Kaggle . . . 654
13.14.1 Obtaining and Organizing the Dataset . . . 655
13.14.2 Image Augmentation . . . 656
13.14.3 Reading the Dataset . . . 657

13.14.4 Defining the Model . . . 657
13.14.5 Defining the Training Functions . . . 658
13.14.6 Training and Validating the Model . . . 659
13.14.7 Classifying the Testing Set and Submitting Results on Kaggle . . . 660

14 Natural Language Processing: Pretraining **663**
14.1 Word Embedding (word2vec) . . . 664
14.1.1 Why Not Use One-hot Vectors? . . . 664
14.1.2 The Skip-Gram Model . . . 664
14.1.3 The Continuous Bag of Words (CBOW) Model . . . 666
14.2 Approximate Training . . . 668
14.2.1 Negative Sampling . . . 669
14.2.2 Hierarchical Softmax . . . 670
14.3 The Dataset for Pretraining Word Embedding . . . 671
14.3.1 Reading and Preprocessing the Dataset . . . 671
14.3.2 Subsampling . . . 672
14.3.3 Loading the Dataset . . . 674
14.3.4 Putting All Things Together . . . 677
14.4 Pretraining word2vec . . . 678
14.4.1 The Skip-Gram Model . . . 679
14.4.2 Training . . . 680
14.4.3 Applying the Word Embedding Model . . . 682
14.5 Word Embedding with Global Vectors (GloVe) . . . 683
14.5.1 The GloVe Model . . . 684
14.5.2 Understanding GloVe from Conditional Probability Ratios . . . 685
14.6 Subword Embedding . . . 686
14.6.1 fastText . . . 686
14.6.2 Byte Pair Encoding . . . 687
14.7 Finding Synonyms and Analogies . . . 690
14.7.1 Using Pretrained Word Vectors . . . 691
14.7.2 Applying Pretrained Word Vectors . . . 692
14.8 Bidirectional Encoder Representations from Transformers (BERT) . . . 695
14.8.1 From Context-Independent to Context-Sensitive . . . 695
14.8.2 From Task-Specific to Task-Agnostic . . . 695
14.8.3 BERT: Combining the Best of Both Worlds . . . 696
14.8.4 Input Representation . . . 697
14.8.5 Pretraining Tasks . . . 699
14.8.6 Putting All Things Together . . . 702
14.9 The Dataset for Pretraining BERT . . . 703
14.9.1 Defining Helper Functions for Pretraining Tasks . . . 704
14.9.2 Transforming Text into the Pretraining Dataset . . . 706
14.10 Pretraining BERT . . . 709
14.10.1 Pretraining BERT . . . 709
14.10.2 Representing Text with BERT . . . 711

15 Natural Language Processing: Applications **715**
15.1 Sentiment Analysis and the Dataset . . . 716
15.1.1 The Sentiment Analysis Dataset . . . 716
15.1.2 Putting All Things Together . . . 719
15.2 Sentiment Analysis: Using Recurrent Neural Networks . . . 720
15.2.1 Using a Recurrent Neural Network Model . . . 720

15.3 Sentiment Analysis: Using Convolutional Neural Networks 723
15.3.1 One-Dimensional Convolutional Layer 724
15.3.2 Max-Over-Time Pooling Layer 726
15.3.3 The TextCNN Model 727
15.4 Natural Language Inference and the Dataset 730
15.4.1 Natural Language Inference 731
15.4.2 The Stanford Natural Language Inference (SNLI) Dataset 731
15.5 Natural Language Inference: Using Attention 735
15.5.1 The Model 736
15.5.2 Training and Evaluating the Model 740
15.6 Fine-Tuning BERT for Sequence-Level and Token-Level Applications 742
15.6.1 Single Text Classification 743
15.6.2 Text Pair Classification or Regression 743
15.6.3 Text Tagging 744
15.6.4 Question Answering 745
15.7 Natural Language Inference: Fine-Tuning BERT 747
15.7.1 Loading Pretrained BERT 748
15.7.2 The Dataset for Fine-Tuning BERT 749
15.7.3 Fine-Tuning BERT 750

16 Recommender Systems **753**
16.1 Overview of Recommender Systems 753
16.1.1 Collaborative Filtering 754
16.1.2 Explicit Feedback and Implicit Feedback 755
16.1.3 Recommendation Tasks 755
16.2 The MovieLens Dataset 756
16.2.1 Getting the Data 756
16.2.2 Statistics of the Dataset 757
16.2.3 Splitting the dataset 758
16.2.4 Loading the data 759
16.3 Matrix Factorization 760
16.3.1 The Matrix Factorization Model 761
16.3.2 Model Implementation 762
16.3.3 Evaluation Measures 762
16.3.4 Training and Evaluating the Model 763
16.4 AutoRec: Rating Prediction with Autoencoders 765
16.4.1 Model 765
16.4.2 Implementing the Model 766
16.4.3 Reimplementing the Evaluator 766
16.4.4 Training and Evaluating the Model 767
16.5 Personalized Ranking for Recommender Systems 768
16.5.1 Bayesian Personalized Ranking Loss and its Implementation 769
16.5.2 Hinge Loss and its Implementation 770
16.6 Neural Collaborative Filtering for Personalized Ranking 771
16.6.1 The NeuMF model 772
16.6.2 Model Implementation 773
16.6.3 Customized Dataset with Negative Sampling 774
16.6.4 Evaluator 774
16.6.5 Training and Evaluating the Model 776
16.7 Sequence-Aware Recommender Systems 778
16.7.1 Model Architectures 778

16.7.2 Model Implementation . . . 780
16.7.3 Sequential Dataset with Negative Sampling . . . 781
16.7.4 Load the MovieLens 100K dataset . . . 782
16.7.5 Train the Model . . . 783
16.8 Feature-Rich Recommender Systems . . . 784
16.8.1 An Online Advertising Dataset . . . 785
16.8.2 Dataset Wrapper . . . 785
16.9 Factorization Machines . . . 787
16.9.1 2-Way Factorization Machines . . . 787
16.9.2 An Efficient Optimization Criterion . . . 788
16.9.3 Model Implementation . . . 788
16.9.4 Load the Advertising Dataset . . . 789
16.9.5 Train the Model . . . 789
16.10 Deep Factorization Machines . . . 790
16.10.1 Model Architectures . . . 791
16.10.2 Implemenation of DeepFM . . . 792
16.10.3 Training and Evaluating the Model . . . 793

17 Generative Adversarial Networks 795
17.1 Generative Adversarial Networks . . . 795
17.1.1 Generate some “real” data . . . 797
17.1.2 Generator . . . 798
17.1.3 Discriminator . . . 798
17.1.4 Training . . . 798
17.2 Deep Convolutional Generative Adversarial Networks . . . 801
17.2.1 The Pokemon Dataset . . . 801
17.2.2 The Generator . . . 802
17.2.3 Discriminator . . . 804
17.2.4 Training . . . 805

18 Appendix: Mathematics for Deep Learning 809
18.1 Geometry and Linear Algebraic Operations . . . 810
18.1.1 Geometry of Vectors . . . 810
18.1.2 Dot Products and Angles . . . 812
18.1.3 Hyperplanes . . . 814
18.1.4 Geometry of Linear Transformations . . . 817
18.1.5 Linear Dependence . . . 819
18.1.6 Rank . . . 819
18.1.7 Invertibility . . . 820
18.1.8 Determinant . . . 821
18.1.9 Tensors and Common Linear Algebra Operations . . . 822
18.2 Eigendecompositions . . . 826
18.2.1 Finding Eigenvalues . . . 826
18.2.2 Decomposing Matrices . . . 827
18.2.3 Operations on Eigendecompositions . . . 827
18.2.4 Eigendecompositions of Symmetric Matrices . . . 828
18.2.5 Gershgorin Circle Theorem . . . 828
18.2.6 A Useful Application: The Growth of Iterated Maps . . . 829
18.2.7 Conclusions . . . 834
18.3 Single Variable Calculus . . . 835
18.3.1 Differential Calculus . . . 835

18.3.2 Rules of Calculus . . . 838
18.4 Multivariable Calculus . . . 845
18.4.1 Higher-Dimensional Differentiation . . . 846
18.4.2 Geometry of Gradients and Gradient Descent . . . 847
18.4.3 A Note on Mathematical Optimization . . . 848
18.4.4 Multivariate Chain Rule . . . 849
18.4.5 The Backpropagation Algorithm . . . 851
18.4.6 Hessians . . . 854
18.4.7 A Little Matrix Calculus . . . 856
18.5 Integral Calculus . . . 861
18.5.1 Geometric Interpretation . . . 861
18.5.2 The Fundamental Theorem of Calculus . . . 863
18.5.3 Change of Variables . . . 865
18.5.4 A Comment on Sign Conventions . . . 866
18.5.5 Multiple Integrals . . . 867
18.5.6 Change of Variables in Multiple Integrals . . . 869
18.6 Random Variables . . . 870
18.6.1 Continuous Random Variables . . . 870
18.7 Maximum Likelihood . . . 887
18.7.1 The Maximum Likelihood Principle . . . 888
18.7.2 Numerical Optimization and the Negative Log-Likelihood . . . 889
18.7.3 Maximum Likelihood for Continuous Variables . . . 891
18.8 Distributions . . . 893
18.8.1 Bernoulli . . . 893
18.8.2 Discrete Uniform . . . 895
18.8.3 Continuous Uniform . . . 896
18.8.4 Binomial . . . 898
18.8.5 Poisson . . . 900
18.8.6 Gaussian . . . 903
18.8.7 Exponential Family . . . 906
18.9 Naive Bayes . . . 907
18.9.1 Optical Character Recognition . . . 908
18.9.2 The Probabilistic Model for Classification . . . 909
18.9.3 The Naive Bayes Classifier . . . 909
18.9.4 Training . . . 910
18.10 Statistics . . . 914
18.10.1 Evaluating and Comparing Estimators . . . 914
18.10.2 Conducting Hypothesis Tests . . . 918
18.10.3 Constructing Confidence Intervals . . . 922
18.11 Information Theory . . . 925
18.11.1 Information . . . 925
18.11.2 Entropy . . . 927
18.11.3 Mutual Information . . . 929
18.11.4 Kullback–Leibler Divergence . . . 933
18.11.5 Cross Entropy . . . 935

19 Appendix: Tools for Deep Learning **939**
19.1 Using Jupyter . . . 939
19.1.1 Editing and Running the Code Locally . . . 939
19.1.2 Advanced Options . . . 943
19.2 Using Amazon SageMaker . . . 944

19.2.1 Registering and Logging In 944
19.2.2 Creating a SageMaker Instance 945
19.2.3 Running and Stopping an Instance 946
19.2.4 Updating Notebooks 947
19.3 Using AWS EC2 Instances 948
19.3.1 Creating and Running an EC2 Instance 948
19.3.2 Installing CUDA 953
19.3.3 Installing MXNet and Downloading the D2L Notebooks 954
19.3.4 Running Jupyter 955
19.3.5 Closing Unused Instances 956
19.4 Using Google Colab 956
19.5 Selecting Servers and GPUs 957
19.5.1 Selecting Servers 958
19.5.2 Selecting GPUs 959
19.6 Contributing to This Book 962
19.6.1 Minor Text Changes 962
19.6.2 Propose a Major Change 962
19.6.3 Adding a New Section or a New Framework Implementation 963
19.6.4 Submitting a Major Change 963
19.7 `d2l` API Document 967

Bibliography **989**

Python Module Index **999**

Index **1001**

Preface

Just a few years ago, there were no legions of deep learning scientists developing intelligent products and services at major companies and startups. When the youngest among us (the authors) entered the field, machine learning did not command headlines in daily newspapers. Our parents had no idea what machine learning was, let alone why we might prefer it to a career in medicine or law. Machine learning was a forward-looking academic discipline with a narrow set of real-world applications. And those applications, e.g., speech recognition and computer vision, required so much domain knowledge that they were often regarded as separate areas entirely for which machine learning was one small component. Neural networks then, the antecedents of the deep learning models that we focus on in this book, were regarded as outmoded tools.

In just the past five years, deep learning has taken the world by surprise, driving rapid progress in fields as diverse as computer vision, natural language processing, automatic speech recognition, reinforcement learning, and statistical modeling. With these advances in hand, we can now build cars that drive themselves with more autonomy than ever before (and less autonomy than some companies might have you believe), smart reply systems that automatically draft the most mundane emails, helping people dig out from oppressively large inboxes, and software agents that dominate the world's best humans at board games like Go, a feat once thought to be decades away. Already, these tools exert ever-wider impacts on industry and society, changing the way movies are made, diseases are diagnosed, and playing a growing role in basic sciences—from astrophysics to biology.

About This Book

This book represents our attempt to make deep learning approachable, teaching you the *concepts*, the *context*, and the *code*.

One Medium Combining Code, Math, and HTML

For any computing technology to reach its full impact, it must be well-understood, well-documented, and supported by mature, well-maintained tools. The key ideas should be clearly distilled, minimizing the onboarding time needing to bring new practitioners up to date. Mature libraries should automate common tasks, and exemplar code should make it easy for practitioners to modify, apply, and extend common applications to suit their needs. Take dynamic web applications as an example. Despite a large number of companies, like Amazon, developing successful database-driven web applications in the 1990s, the potential of this technology to aid creative entrepreneurs has been realized to a far greater degree in the past ten years, owing in part to the development of powerful, well-documented frameworks.

Testing the potential of deep learning presents unique challenges because any single application brings together various disciplines. Applying deep learning requires simultaneously understanding (i) the motivations for casting a problem in a particular way; (ii) the mathematics of a given modeling approach; (iii) the optimization algorithms for fitting the models to data; and (iv) the engineering required to train models efficiently, navigating the pitfalls of numerical computing and getting the most out of available hardware. Teaching both the critical thinking skills required to formulate problems, the mathematics to solve them, and the software tools to implement those solutions all in one place presents formidable challenges. Our goal in this book is to present a unified resource to bring would-be practitioners up to speed.

At the time we started this book project, there were no resources that simultaneously (i) were up to date; (ii) covered the full breadth of modern machine learning with substantial technical depth; and (iii) interleaved exposition of the quality one expects from an engaging textbook with the clean runnable code that one expects to find in hands-on tutorials. We found plenty of code examples for how to use a given deep learning framework (e.g., how to do basic numerical computing with matrices in TensorFlow) or for implementing particular techniques (e.g., code snippets for LeNet, AlexNet, ResNets, etc) scattered across various blog posts and GitHub repositories. However, these examples typically focused on *how* to implement a given approach, but left out the discussion of *why* certain algorithmic decisions are made. While some interactive resources have popped up sporadically to address a particular topic, e.g., the engaging blog posts published on the website Distill[3], or personal blogs, they only covered selected topics in deep learning, and often lacked associated code. On the other hand, while several textbooks have emerged, most notably (Goodfellow et al., 2016), which offers a comprehensive survey of the concepts behind deep learning, these resources do not marry the descriptions to realizations of the concepts in code, sometimes leaving readers clueless as to how to implement them. Moreover, too many resources are hidden behind the paywalls of commercial course providers.

We set out to create a resource that could (i) be freely available for everyone; (ii) offer sufficient technical depth to provide a starting point on the path to actually becoming an applied machine learning scientist; (iii) include runnable code, showing readers *how* to solve problems in practice; (iv) allow for rapid updates, both by us and also by the community at large; and (v) be complemented by a forum[4] for interactive discussion of technical details and to answer questions.

These goals were often in conflict. Equations, theorems, and citations are best managed and laid out in LaTeX. Code is best described in Python. And webpages are native in HTML and JavaScript. Furthermore, we want the content to be accessible both as executable code, as a physical book, as a downloadable PDF, and on the Internet as a website. At present there exist no tools and no workflow perfectly suited to these demands, so we had to assemble our own. We describe our approach in detail in Section 19.6. We settled on GitHub to share the source and to allow for edits, Jupyter notebooks for mixing code, equations and text, Sphinx as a rendering engine to generate multiple outputs, and Discourse for the forum. While our system is not yet perfect, these choices provide a good compromise among the competing concerns. We believe that this might be the first book published using such an integrated workflow.

[3] http://distill.pub
[4] http://discuss.d2l.ai

Learning by Doing

Many textbooks teach a series of topics, each in exhaustive detail. For example, Chris Bishop's excellent textbook (Bishop, 2006), teaches each topic so thoroughly, that getting to the chapter on linear regression requires a non-trivial amount of work. While experts love this book precisely for its thoroughness, for beginners, this property limits its usefulness as an introductory text.

In this book, we will teach most concepts *just in time*. In other words, you will learn concepts at the very moment that they are needed to accomplish some practical end. While we take some time at the outset to teach fundamental preliminaries, like linear algebra and probability, we want you to taste the satisfaction of training your first model before worrying about more esoteric probability distributions.

Aside from a few preliminary notebooks that provide a crash course in the basic mathematical background, each subsequent chapter introduces both a reasonable number of new concepts and provides single self-contained working examples—using real datasets. This presents an organizational challenge. Some models might logically be grouped together in a single notebook. And some ideas might be best taught by executing several models in succession. On the other hand, there is a big advantage to adhering to a policy of *one working example, one notebook*: This makes it as easy as possible for you to start your own research projects by leveraging our code. Just copy a notebook and start modifying it.

We will interleave the runnable code with background material as needed. In general, we will often err on the side of making tools available before explaining them fully (and we will follow up by explaining the background later). For instance, we might use *stochastic gradient descent* before fully explaining why it is useful or why it works. This helps to give practitioners the necessary ammunition to solve problems quickly, at the expense of requiring the reader to trust us with some curatorial decisions.

This book will teach deep learning concepts from scratch. Sometimes, we want to delve into fine details about the models that would typically be hidden from the user by deep learning frameworks' advanced abstractions. This comes up especially in the basic tutorials, where we want you to understand everything that happens in a given layer or optimizer. In these cases, we will often present two versions of the example: one where we implement everything from scratch, relying only on the NumPy interface and automatic differentiation, and another, more practical example, where we write succinct code using high-level APIs of deep learning frameworks. Once we have taught you how some component works, we can just use the high-level APIs in subsequent tutorials.

Content and Structure

The book can be roughly divided into three parts, which are presented by different colors in Fig. 1:

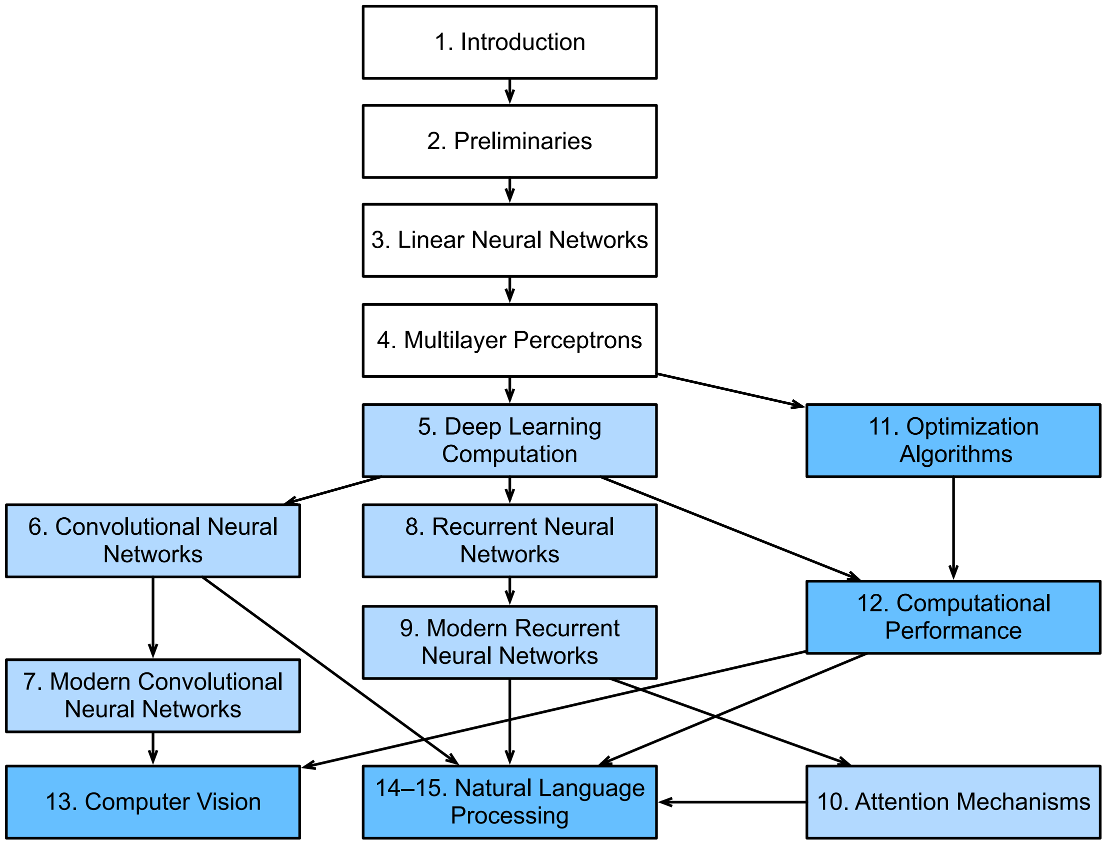
1. Introduction
2. Preliminaries
3. Linear Neural Networks
4. Multilayer Perceptrons
5. Deep Learning Computation
11. Optimization Algorithms
6. Convolutional Neural Networks
8. Recurrent Neural Networks
12. Computational Performance
9. Modern Recurrent Neural Networks
7. Modern Convolutional Neural Networks
13. Computer Vision
14–15. Natural Language Processing
10. Attention Mechanisms

Fig. 1: Book structure

- The first part covers basics and preliminaries. Chapter 1 offers an introduction to deep learning. Then, in Chapter 2, we quickly bring you up to speed on the prerequisites required for hands-on deep learning, such as how to store and manipulate data, and how to apply various numerical operations based on basic concepts from linear algebra, calculus, and probability. Chapter 3 and Chapter 4 cover the most basic concepts and techniques of deep learning, such as linear regression, multilayer perceptrons and regularization.
- The next five chapters focus on modern deep learning techniques. Chapter 5 describes the various key components of deep learning calculations and lays the groundwork for us to subsequently implement more complex models. Next, in Chapter 6 and Chapter 7, we introduce convolutional neural networks (CNNs), powerful tools that form the backbone of most modern computer vision systems. Subsequently, in Chapter 8 and Chapter 9, we introduce recurrent neural networks (RNNs), models that exploit temporal or sequential structure in data, and are commonly used for natural language processing and time series prediction. In Chapter 10, we introduce a new class of models that employ a technique called attention mechanisms and they have recently begun to displace RNNs in natural language processing. These sections will get you up to speed on the basic tools behind most modern applications of deep learning.
- Part three discusses scalability, efficiency, and applications. First, in Chapter 11, we discuss several common optimization algorithms used to train deep learning models. The next

chapter, Chapter 12 examines several key factors that influence the computational performance of your deep learning code. In Chapter 13, we illustrate major applications of deep learning in computer vision. In Chapter 14 and Chapter 15, we show how to pretrain language representation models and apply them to natural language processing tasks.

Code

Most sections of this book feature executable code because of our belief in the importance of an interactive learning experience in deep learning. At present, certain intuitions can only be developed through trial and error, tweaking the code in small ways and observing the results. Ideally, an elegant mathematical theory might tell us precisely how to tweak our code to achieve a desired result. Unfortunately, at present, such elegant theories elude us. Despite our best attempts, formal explanations for various techniques are still lacking, both because the mathematics to characterize these models can be so difficult and also because serious inquiry on these topics has only just recently kicked into high gear. We are hopeful that as the theory of deep learning progresses, future editions of this book will be able to provide insights in places the present edition cannot.

At times, to avoid unnecessary repetition, we encapsulate the frequently-imported and referred-to functions, classes, etc. in this book in the `d2l` package. For any block such as a function, a class, or multiple imports to be saved in the package, we will mark it with `#@save`. We offer a detailed overview of these functions and classes in Section 19.7. The `d2l` package is light-weight and only requires the following packages and modules as dependencies:

```python
#@save
import collections
from collections import defaultdict
from IPython import display
import math
from matplotlib import pyplot as plt
import os
import pandas as pd
import random
import re
import shutil
import sys
import tarfile
import time
import requests
import zipfile
import hashlib
d2l = sys.modules[__name__]
```

Most of the code in this book is based on Apache MXNet. MXNet is an open-source framework for deep learning and the preferred choice of AWS (Amazon Web Services), as well as many colleges and companies. All of the code in this book has passed tests under the newest MXNet version. However, due to the rapid development of deep learning, some code *in the print edition* may not work properly in future versions of MXNet. However, we plan to keep the online version up-to-date. In case you encounter any such problems, please consult *Installation* (page 9) to update your code and runtime environment.

Here is how we import modules from MXNet.

```
#@save
from mxnet import autograd, context, gluon, image, init, np, npx
from mxnet.gluon import nn, rnn
```

Target Audience

This book is for students (undergraduate or graduate), engineers, and researchers, who seek a solid grasp of the practical techniques of deep learning. Because we explain every concept from scratch, no previous background in deep learning or machine learning is required. Fully explaining the methods of deep learning requires some mathematics and programming, but we will only assume that you come in with some basics, including (the very basics of) linear algebra, calculus, probability, and Python programming. Moreover, in the Appendix, we provide a refresher on most of the mathematics covered in this book. Most of the time, we will prioritize intuition and ideas over mathematical rigor. There are many terrific books which can lead the interested reader further. For instance, Linear Analysis by Bela Bollobas (Bollobas, 1999) covers linear algebra and functional analysis in great depth. All of Statistics (Wasserman, 2013) is a terrific guide to statistics. And if you have not used Python before, you may want to peruse this Python tutorial[5].

Forum

Associated with this book, we have launched a discussion forum, located at discuss.d2l.ai[6]. When you have questions on any section of the book, you can find the associated discussion page link at the end of each chapter.

Acknowledgments

We are indebted to the hundreds of contributors for both the English and the Chinese drafts. They helped improve the content and offered valuable feedback. Specifically, we thank every contributor of this English draft for making it better for everyone. Their GitHub IDs or names are (in no particular order): alxnorden, avinashingit, bowen0701, brettkoonce, Chaitanya Prakash Bapat, cryptonaut, Davide Fiocco, edgarroman, gkutiel, John Mitro, Liang Pu, Rahul Agarwal, Mohamed Ali Jamaoui, Michael (Stu) Stewart, Mike Müller, NRauschmayr, Prakhar Srivastav, sad-, sfermigier, Sheng Zha, sundeepteki, topecongiro, tpdi, vermicelli, Vishaal Kapoor, Vishwesh Ravi Shrimali, YaYaB, Yuhong Chen, Evgeniy Smirnov, lgov, Simon Corston-Oliver, Igor Dzreyev, Ha Nguyen, pmuens, Andrei Lukovenko, senorcinco, vfdev-5, dsweet, Mohammad Mahdi Rahimi, Abhishek Gupta, uwsd, DomKM, Lisa Oakley, Bowen Li, Aarush Ahuja, Prasanth Buddareddygari, brianhendee, mani2106, mtn, lkevinzc, caojilin, Lakshya, Fiete Lüer, Surbhi Vijayvargeeya, Muhyun Kim, dennismalmgren, adursun, Anirudh Dagar, liqingnz, Pedro Larroy, lgov, ati-ozgur, Jun Wu, Matthias Blume, Lin Yuan, geogunow, Josh Gardner, Maximilian Böther, Rakib Islam, Leonard Lausen, Abhinav Upadhyay, rongruosong, Steve Sedlmeyer, Ruslan Baratov, Rafael Schlatter, liusy182, Giannis Pappas, ati-ozgur, qbaza, dchoi77, Adam Gerson, Phuc Le, Mark Atwood, christabella, vn09, Haibin Lin, jjangga0214, RichyChen, noelo, hansent, Giel Dops, dvincent1337, WhiteD3vil, Peter Kulits, codypenta, joseppinilla, ahmaurya, karolszk, heytitle, Peter Goetz, rigtorp, Tiep Vu, sfilip, mlxd, Kale-ab Tessera, Sanjar Adilov,

[5] http://learnpython.org/
[6] https://discuss.d2l.ai/

MatteoFerrara, hsneto, Katarzyna Biesialska, Gregory Bruss, Duy–Thanh Doan, paulaurel, graytowne, Duc Pham, sl7423, Jaedong Hwang, Yida Wang, cys4, clhm, Jean Kaddour, austinmw, trebeljahr, tbaums, Cuong V. Nguyen, pavelkomarov, vzlamal, NotAnotherSystem, J-Arun-Mani, jancio, eldarkurtic, the-great-shazbot, doctorcolossus, gducharme, cclauss, Daniel-Mietchen, hoonose, biagiom, abhinavsp0730, jonathanhrandall, ysraell, Nodar Okroshiashvili, UgurKap, Jiyang Kang, StevenJokes, Tomer Kaftan, liweiwp, netyster, ypandya, NishantTharani, heiligerl, SportsTHU, Hoa Nguyen, manuel-arno-korfmann-webentwicklung, aterzis-personal, nxby, Xiaoting He, Josiah Yoder, mathresearch, mzz2017, jroberayalas, iluu, ghejc, BSharmi, vkramdev, simonwardjones, LakshKD, TalNeoran, djliden, Nikhil95, Oren Barkan, guoweis, haozhu233, pratikhack, 315930399, tayfunural, steinsag, charleybeller, Andrew Lumsdaine, Jiekui Zhang, Deepak Pathak, Florian Donhauser, Tim Gates, Adriaan Tijsseling, Ron Medina, Gaurav Saha, Murat Semerci, Lei Mao[7].

We thank Amazon Web Services, especially Swami Sivasubramanian, Raju Gulabani, Charlie Bell, and Andrew Jassy for their generous support in writing this book. Without the available time, resources, discussions with colleagues, and continuous encouragement this book would not have happened.

Summary

- Deep learning has revolutionized pattern recognition, introducing technology that now powers a wide range of technologies, including computer vision, natural language processing, automatic speech recognition.
- To successfully apply deep learning, you must understand how to cast a problem, the mathematics of modeling, the algorithms for fitting your models to data, and the engineering techniques to implement it all.
- This book presents a comprehensive resource, including prose, figures, mathematics, and code, all in one place.
- To answer questions related to this book, visit our forum at https://discuss.d2l.ai/.
- All notebooks are available for download on GitHub.

Exercises

1. Register an account on the discussion forum of this book discuss.d2l.ai[8].
2. Install Python on your computer.
3. Follow the links at the bottom of the section to the forum, where you will be able to seek out help and discuss the book and find answers to your questions by engaging the authors and broader community.

Discussions[9]

[7] https://github.com/leimao
[8] https://discuss.d2l.ai/
[9] https://discuss.d2l.ai/t/18