Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Progetto Finale

Flowers Recognition

Autore:
Alessandro Capelli - 816302 - a.capelli11@campus.unimib.it

21 gennaio 2021

Sommario

L'elaborato ha lo scopo di presentare un caso applicativo di CNN che sfrutta l'efficacia del transfer learning per l'immagazzinamento di conoscenza che può essere utilizzata in un task differente da quello da cui è stata ricavata. Tramite una serie di risultati empirici è possibile osservare l'ottimo livello di accuratezza raggiunto sul test set a fronte di differenti architetture di base prese in considerazione. ResNet50V2 si è dimostrata la rete più abile a generalizzare il problema di classificazione sottoposto, anche utilizzando un unico layer trainabile. Ottimizzando la struttura e gli iperparametri del modello è stato possibile raggiungere il 95% di accuratezza sul test set. Successivamente è stata applicata la tecnica di pruning per la compressione del modello mantenendo un livello di accuratezza accettabile (88%).

1 Introduzione

Il problema scelto per l'elaborazione del progetto in questione consiste nella classificazione di immagini contenenti diverse specie di fiori. In particolare, il problema consiste in una classificazione in 102 categorie di una serie di immagini a colori (RGB) in formato jpg raffiguranti dei fiori in diverse angolature, distanze e colori. L'obiettivo dell'elaborato consiste nella valutazione empirica dell'efficacia derivante dall'applicazione della tecnica di *transfer learning* [1] al dataset selezionato per dimostrare la reale possibilità di utilizzo di modelli pre-allenati per task differenti da quelli per cui è stata allenata originariamente la rete. Tale problema è di particolare interesse in quanto ritrae una problematica ampia all'interno dell'apprendimento automatico, ovvero la possibilità di allenare modelli "generali" che possano poi essere specializzati su compiti diversi, creando reti capaci di immagazzinare conoscenza cogliendo la struttura sottostante i vari tipi di dataset (e. g., la capacità di riconoscere le linee in un'immagine può essere sfruttata in contesti molto differenti). Gli esperimenti effettuati sono stati condotti con l'intento di trovare la migliore struttura della rete in relazione al dataset e per testare a vari livelli di granularità gli iperparametri più idonei; in particolare, sono stati seguiti due differenti approcci per l'ottimizzazione degli stessi: inizialmente è stato scelto l'utilizzo di una procedura "manuale" per effettuare un'analisi esplorativa delle architetture e di alcuni iperparametri, successivamente, per trovare la migliore combinazione dei valori, è stata adottata una tecnica automatica basata su *RandomSearch*. Infine è stata applicata una tecnica

per la compressione del miglior modello ottenuto chiamata *pruning* [2]. Un fattore da considerare è stato il costo computazionale richiesto per il training dei modelli: data la grossa mole di dati presenti nel dataset, l'allenamento di ogni modello ha richiesto un tempo di calcolo non indifferente (variabile in base alle risorse momentaneamente disponibili offerte da Google Colab) che ha portato alla ricerca di piattaforme di calcolo alternative (e. g., vast.ai [3]), all'utilizzo di macchine in parallelo (comprese macchine locali che hanno mostrato grossi problemi di elaborazione dettati dalla mancanza di GPU dedicate) e all'ipotesi di iscrizione al servizio Google Colab Pro (ipotesi non sviluppata in quanto tale servizio è attualmente disponibile solamente negli Stati Uniti ed in Canada). Tali limiti hanno dettato una scelta più consapevole e mirata dei test effettuati, ed al contempo limitante in termini di esplorazione del campo degli iperparametri che inizialmente è stato prefissato per l'ottenimento di un modello più ottimizzato. È necessario specificare che tutti gli esperimenti effettuati sono riproducibili grazie alla specifica di un seed che permette di poter osservare ed eseguire da chiunque il codice, ottenendo i medesimi risultati.

2 Dataset

Il dataset è chiamato *"Oxford Flower Dataset"* [4] ed è composto da un totale di 8189 immagini raccolte sul Web tramite tool di scraping e divise in 102 differenti categorie. La distribuzione iniziale delle immagini per ogni categoria è mostrata in figura 1.

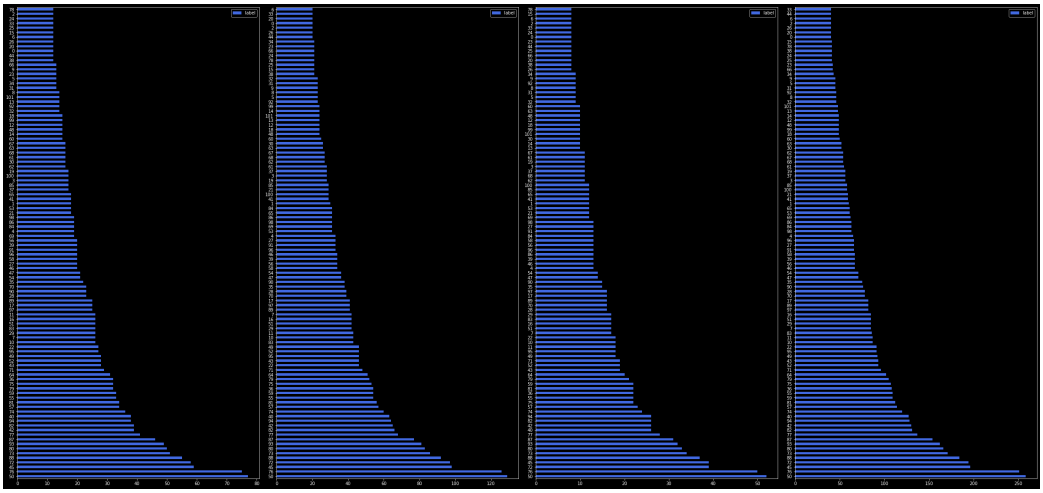

Figura 1: Distribuzione delle immagini per ogni categoria ordinate in modo crescente. Da sinistra: dataset iniziale, train set, validation set, test set.

Per lo sviluppo dei modelli di deep learning è stato scelto di suddividere il dataset iniziale in tre differenti dataset (train set, validation set e test set), utilizzando proporzioni differenti (gli elementi sono rispettivamente 4094, 1638 e 2457), scelte prendendo spunto dall'attuale stato dell'arte trovato nella letteratura [5]. Un aspetto importante è stato quello di scegliere se mantenere o meno la "stratificazione" delle differenti label associate alle categorie: la scelta finale consiste nel mantenimento di tali proporzioni tra i diversi dataset, così da cogliere relazioni "a priori" della distribuzione degli eventi sottostante che si pensa possa rappresentare meglio il caso reale. Per mediare l'eventuale overfitting dettato da tale scelta, è stata successivamente applicata la tecnica di *augmentation* del dataset in modo granulare per i differenti dataset considerati, utilizzando le immagini in ordine casuale tramite shuffle (esempio in figura 2).

Figura 2: Esempio di immagini estratte dal train set aumentato.

3 Approccio metodologico

Per la creazione del modello finale, sono stati condotti una serie di esperimenti elencati di seguito, aventi l'obiettivo di trovare la migliore configurazione in termini di architettura e di iperparametri per massimizzare l'accuratezza sui dati di validazione (metrica utilizzata come riferimento). Le architetture utilizzate negli esperimenti sono:

- **singolo layer denso:** architettura base, 1 layer denso
  - BASE MODEL
  - Flatten
  - Dense: *units=102, activation=softmax*
- **estesa:** architettura base, 1 max-pooling, 2 layer densi, 1 dropout
  - BASE MODEL
  - MaxPooling2D: *window size=(2, 2)*
  - Flatten
  - Dense: NUM UNITS, ACTIVATION, REGULARIZER
  - Dropout: DROPOUT RATE
  - Dense: *units=102, activation=softmax*
- **fine tuning:**

  - dallo strato X: indica un'architettura sulla quale viene effettuato il fine tuning della rete dallo strato $\frac{numero\ di\ layers}{X}$ in poi (i precedenti rimangono non trainabili)
  - di tutta la rete: indica un'architettura sulla quale viene effettuato il fine tuning dell'intera rete

- **glorot:** indica un'architettura in cui i pesi vengono inizializzati utilizzando la distribuzione *"Glorot"*. Viene utilizzata come riferimento per verificare l'efficacia del transfer learning

Si noti che le diciture in maiuscolo indicano che tale valore viene fatto variare durante i test eseguiti.

Tutti gli esperimenti hanno alcune caratteristiche in comune riportate di seguito:

- la dimensione del batch size è pari a 32 (è stata provata anche una dimensione pari a 16)
- è stata introdotta la tecnica di regolarizzazione chiamata *early stopping* con una *patience* pari a 5 per evitare overfitting
- dopo ogni epoca viene effettuato un salvataggio (*checkpoint*) del modello
- la funzione di loss utilizzata è la *categorical cross-entropy*
- è stato utilizzato l'ottimizzatore *Adam* i cui parametri (LEARNING RATE, BETA 1 e BETA 2) sono stati fatti variare per adattarsi ai dati
- sono state scelte le seguenti metriche [6]: *accuracy* (scelta come metrica di riferimento), *precision*, *recall*, *f1-score*, *AUC*

Inizialmente è stato utilizzato un approccio "manuale" per la scelta degli iperparametri e per decidere quali tecniche di regolarizzazione applicare, così da effettuare un'analisi esplorativa e ridurre il dominio della ricerca dei valori; successivamente ci si è affidati alla libreria di tuning *"kerastuner"* che permette l'applicazione di *RandomSearch* per la ricerca dei migliori iperparametri in modo "automatico". Una volta scelto il modello finale e testato lo stesso sul test set, è stato quantificato il vantaggio espresso dall'utilizzo dei

pesi del modello pre-allenato, ovvero del transfer learning, rispetto all'utilizzo della medesima architettura con i pesi inizializzati secondo la distribuzione Glorot; infine è stata applicata una tecnica di compressione chiamata *pruning* che ha permesso di passare ad un modello sparso, il quale è meno efficiente in termini di velocità (effettuando calcoli su matrici sparse) rispetto alla versione densa ma che ha permesso una compressione in termini di pesi e conseguentemente di spazio di occupazione richiesto. È necessario precisare che per effettuare l'operazione di pruning è stata utilizzata la libreria *"tensorflow-model-optimization"* [7] che mette a disposizione strumenti per l'ottimizzazione dei modelli per la distribuzione e l'esecuzione.

4 Risultati

4.1 Esperimento #1

Il primo esperimento condotto ha lo scopo di determinare quale tra i modelli *pre-trained* scelti si adatta meglio al dataset. I modelli testati sono i seguenti:

- **VGG16 [8]:** ha una dimensione di 528 MB, un numero di parametri pari a 138 357 544 - 14 714 688 (senza strati densi) e una profondità pari a 19 (senza strati densi)
- **XCEPTION [9]:** ha una dimensione di 88 MB, un numero di parametri pari a 22 910 480 - 20 861 480 (senza strati densi) e una profondità pari a 132 (senza strati densi)
- **RESNET50V2 [10]:** ha una dimensione di 98 MB, un numero di parametri pari a 25 613 800 - 23 564 800 (senza strati densi) e una profondità pari a 190 (senza strati densi)

Tali modelli sono stati testati con l'architettura **estesa** (trainando solo i pesi dei layer densi), utilizzando un dropout rate pari a 0.2 (e 0.3 con batch size pari a 16) e la funzione di attivazione 'elu' nello strato denso. Sono stati fatti variare il learning rate (con valori di 0.001 e 0.0001) ed il numero di unità dello strato denso (con valori di 128, 256 e 512). Utilizzando come modello base RESNET50V2, è stato possibile ottenere un valore di accuratezza sul training set pari a 0.9 e di 0.87 per quanto riguarda il validation set. Osservando gli ottimi risultati (figura 3) ottenuti già a fronte del primo esperimento, è stato deciso di proseguire utilizzando RESNET50V2.

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	VAL LOSS	VAL ACCURACY
5	16	VGG16	0.2	0.001	256	elu	None	1,24	0,66	0,96	0,74
5	16	VGG16	0.3	0.001	256	elu	None	1,38	0,61	1,01	0,73
5	16	XCEPTION	0.2	0.001	256	elu	None	1,61	0,57	1,25	0,67
5	16	XCEPTION	0.3	0.001	256	elu	None	2,06	0,47	1,42	0,62
5	16	RESNET50V2	0.2	0.001	256	elu	None	1,91	0,52	1,23	0,68
5	16	RESNET50V2	0.3	0.001	256	elu	None	2,72	0,36	1,76	0,56
10	32	VGG16	0.2	0.001	128	elu	None	0,92	0,74	0,79	0,77
10	32	VGG16	0.2	0.001	256	elu	None	0,8	0,77	0,76	0,77
10	32	VGG16	0.2	0.001	512	elu	None	0,79	0,77	0,85	0,77
10	32	VGG16	0.2	0.0001	128	elu	None	1,98	0,53	1,61	0,65
10	32	VGG16	0.2	0.0001	256	elu	None	1,51	0,64	1,24	0,71
10	32	VGG16	0.2	0.0001	512	elu	None	1,17	0,71	1	0,75
10	32	XCEPTION	0.2	0.001	128	elu	None	1,83	0,49	1,35	0,64
10	32	XCEPTION	0.2	0.001	256	elu	None	1,37	0,62	1,13	0,7
10	32	XCEPTION	0.2	0.001	512	elu	None	1,08	0,71	1,05	0,75
10	32	XCEPTION	0.2	0.0001	128	elu	None	0,53	0,85	0,61	0,83
10	32	XCEPTION	0.2	0.0001	256	elu	None	0,4	0,88	0,64	0,83
10	32	XCEPTION	0.2	0.0001	512	elu	None	0,32	0,9	0,65	0,83
10	32	RESNET50V2	0.2	0.001	128	elu	None	1,99	0,49	1,51	0,62
10	32	RESNET50V2	0.2	0.001	256	elu	None	1,44	0,63	1,1	0,72
10	32	RESNET50V2	0.2	0.001	512	elu	None	1,17	0,71	1,02	0,77
10	32	RESNET50V2	0.2	0.0001	128	elu	None	0,57	0,84	0,65	0,84
10	32	RESNET50V2	0.2	0.0001	256	elu	None	0,45	0,88	0,54	0,87
10	32	RESNET50V2	0.2	0.0001	512	elu	None	0,36	0,9	0,52	0,87

Figura 3: Risultati esperimento 1

4.2 Esperimento #2

Durante il secondo esperimento è stata effettuata un'analisi più approfondita delle tecniche di regolarizzazione utilizzate: è stata testata l'architettura a **singolo layer denso** (senza tecniche di regolarizzazione) a confronto con quella **estesa** facendo variare leggermente il dropout rate, utilizzando un learning rate pari a 0.0001 (che è risultato vincente rispetto ad un valore più grande) ed introducendo la tecnica di regolarizzazione *"weight decay"* (sia L1 che L2). I risultati, riportati in figura 4 (architettura estesa) e figura 5 (architettura a singolo layer denso), non mostrano grosse differenze con i risultati precedenti (se non per quanto riguarda l'accuratezza ottenuta con l'architettura a singolo layer denso sul training set) ma è necessario prestare attenzione al valore della somma dei pesi, che viene ridotto drasticamente grazie all'utilizzo di L1 ed L2 (architettura estesa) e porta ad una generalizzazione teorica maggiore su nuovi dati rispetto all'utilizzo dell'architettura a singolo layer denso in cui si osserva l'effetto dell'overfitting.

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
15	32	RESNET50V2	0,25	0,0001	256	elu	L1(0.005)	7,92	0,66	0,86	0,48	0,61	7,46	0,75	0,88	0,59	0,7
15	32	RESNET50V2	0,25	0,0001	256	elu	L2(0.01)	1,99	0,89	0,92	0,87	0,89	2,07	0,87	0,9	0,84	0,87
15	32	RESNET50V2	0,25	0,0001	256	elu	L1(0.0005)	6,52	0,85	0,91	0,81	0,86	6,27	0,85	0,9	0,82	0,86
15	32	RESNET50V2	0,25	0,0001	256	elu	L2(0.001)	0,75	0,9	0,93	0,89	0,91	0,89	0,88	0,9	0,86	0,88
15	32	RESNET50V2	0,2	0,0001	256	elu	L2(0.01)	1,85	0,91	0,94	0,89	0,92	2,01	0,87	0,91	0,85	0,88
15	32	RESNET50V2	0,25	0,0001	256	elu	L2(0.01)	2	0,9	0,93	0,87	0,9	2,1	0,87	0,9	0,85	0,87
15	32	RESNET50V2	0,3	0,0001	256	elu	L2(0.01)	2,09	0,87	0,92	0,85	0,88	2,2	0,86	0,88	0,84	0,86
15	32	RESNET50V2	0,25	0,0001	256	relu	L2(0.01)	1,73	0,87	0,92	0,84	0,88	1,75	0,87	0,92	0,84	0,87

Figura 4: Risultati esperimento 2: architettura estesa

EPOCHS	BATCH SIZE	BASE MODEL	LR	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,0001	0,24	0,94	0,94	0,93	0,94	0,67	0,87	0,88	0,86	0,87
10	32	RESNET50V2	0,0001	0,22	0,94	0,95	0,94	0,94	0,7	0,86	0,88	0,86	0,87

Figura 5: Risultati esperimento 2: architettura a singolo layer denso

4.3 Esperimento #3

Il successivo esperimento ha l'obiettivo di individuare il punto migliore da cui effettuare il fine tuning dei pesi. In particolare, sono stati individuati due punti:

- $X = 2$: viene effettuato il fine tuning dalla metà dell'architettura in poi. I parametri da allenare sono circa 35 milioni per quanto riguarda l'architettura a singolo layer denso (figura 6) e circa 30 milioni per l'architettura estesa (figura 7)
- $X = 1.05$: viene effettuato il fine tuning dagli ultimi strati in avanti. I parametri da allenare sono circa 17 milioni per quanto riguarda l'architettura a singolo layer denso (figura 8) e circa 12 milioni per l'architettura estesa (figura 9)

EPOCHS	BATCH SIZE	BASE MODEL	LR	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,0001	0,07	0,98	0,98	0,97	0,98	0,33	0,92	0,93	0,92	0,92

Figura 6: Risultati esperimento 3: architettura a singolo layer denso con fine tuning a 2

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,5	0,0001	256	elu	L2(0.2)	0,94	0,95	0,97	0,92	0,95	1	0,93	0,96	0,92	0,94

Figura 7: Risultati esperimento 3: architettura estesa con fine tuning a 2

EPOCHS	BATCH SIZE	BASE MODEL	LR	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,0001	0,1	0,97	0,97	0,96	0,97	0,53	0,88	0,89	0,87	0,88
10	32	RESNET50V2	0,0001	0,08	0,97	0,98	0,97	0,97	0,43	0,91	0,92	0,91	0,91

Figura 8: Risultati esperimento 3: architettura a singolo layer denso con fine tuning a 1.05

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,5	0,0001	256	elu	L2(0.2)	2,35	0,86	0,94	0,78	0,85	2,19	0,87	0,93	0,83	0,88

Figura 9: Risultati esperimento 3: architettura estesa con fine tuning a 1.05

Nel medesimo esperimento si è provato ad eseguire il fine tuning sull'intera rete, sia per quanto riguarda l'architettura a singolo layer denso (figura 10), sia per l'architettura estesa (figura 11) in cui vengono allenati circa 32 milioni di parametri.

EPOCHS	BATCH SIZE	BASE MODEL	LR	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,0001	0,12	0,96	0,97	0,95	0,96	0,47	0,9	0,91	0,89	0,9

Figura 10: Risultati esperimento 3: architettura a singolo layer denso con fine tuning su tutta la rete

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,2	0,0001	256	elu	L2(0.2)	0,95	0,95	0,97	0,93	0,95	1,03	0,91	0,94	0,89	0,92
10	32	RESNET50V2	0,2	0,0001	256	elu	L2(0.3)	0,66	0,96	0,97	0,94	0,95	0,75	0,93	0,95	0,91	0,93
10	32	RESNET50V2	0,4	0,0001	256	elu	L2(0.2)	0,51	0,97	0,98	0,96	0,97	0,73	0,93	0,95	0,91	0,93
10	32	RESNET50V2	0,4	0,0001	256	elu	L2(0.3)	0,55	0,98	0,98	0,96	0,97	0,75	0,93	0,95	0,92	0,93

Figura 11: Risultati esperimento 3: architettura estesa con fine tuning su tutta la rete

I migliori risultati (figure 12 e 13) si osservano nell'ultimo esperimento effettuato, ovvero effettuando il **fine tuning** su tutta la rete **estesa**: è possibile raggiungere un'accuratezza del 99% sul training set, del 95% sul validation set e del **94,5% sul test set**. Per questo motivo, tale architettura è stata considerata quella ottimale e quindi presa come riferimento per le successive analisi.

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
50	32	RESNET50V2	0,45	0,00002	256	elu	L2(0.3)	0,33	0,99	0,99	0,98	0,99	0,46	0,95	0,97	0,94	0,96

Figura 12: Risultati esperimento 3: architettura estesa con fine tuning su tutta la rete e con iperparametri ottimizzati

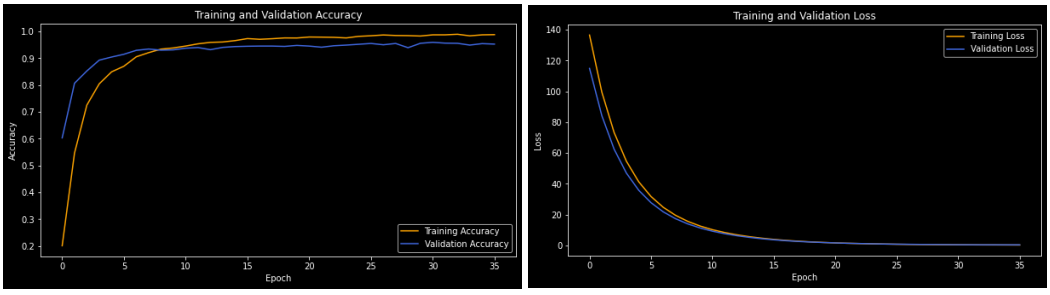

Figura 13: Da sinistra: accuratezza e loss dell'architettura estesa con fine tuning su tutta la rete e con iperparametri ottimizzati

4.4 Esperimento #4

L'obiettivo di questo esperimento è stato quello di valutare l'efficacia derivante dall'utilizzo di un modello pre-allenato. La valutazione mette a confronto l'ultimo modello ottenuto, con il medesimo inizializzato con la distribuzione Glorot. I risultati ottenuti dopo aver inizializzato i pesi in modo randomico seguendo tale distribuzione sono visibili nelle figure 14 (partendo dall'architettura a singolo layer denso) e 15 (partendo dall'architettura estesa).

EPOCHS	BATCH SIZE	BASE MODEL	LR	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,0001	2,19	0,42	0,7	0,25	0,36	2,19	0,45	0,7	0,29	0,41

Figura 14: Risultati esperimento 4: architettura a singolo layer denso inizializzata con Glorot

EPOCHS	BATCH SIZE	BASE MODEL	DROPOUT RATE	LR	NUM UNITS	ACTIVATION	REGULARIZER	LOSS	ACCURACY	PRECISION	RECALL	F1-SCORE	VAL LOSS	VAL ACCURACY	VAL PRECISION	VAL RECALL	VAL F1-SCORE
10	32	RESNET50V2	0,4	0,0001	256	elu	L2(0.3)	4,31	0,14	0,23	0,01	0,02	3,97	0,22	0	0	0

Figura 15: Risultati esperimento 4: architettura estesa inizializzata con Glorot

4.5 Esperimento #5

Dopo aver effettuato un'ottimizzazione manuale degli iperparametri e scelto l'architettura e le tecniche di regolarizzazione da utilizzare, sono stati ottimizzati gli iperparametri restati in modo automatico. I migliori cinque risultati ottenuti (a fronte di circa cinquanta test condotti) sono riportati nella figura 16. Utilizzando un dropout rate elevato (maggiore di 0,5), un learning rate di $2e-5$, un numero elevato di unità nello strato denso e la funzione di attivazione 'elu', l'accuratezza sul validation set ha raggiunto il 97%.

DROPOUT RATE	LR	NUM UNITS	BETA 1	BETA 2	ACCURACY
0,6	2,00E-05	896	0,95	0,9987	0,97
0,7	2,00E-05	896	0,9	0,9991	0,97
0,6	2,00E-05	1024	0,95	0,9999	0,97
0,7	2,00E-05	768	0,95	0,9991	0,97
0,5	2,00E-05	768	0,9	0,9987	0,97

Figura 16: Risultati esperimento 5: tuning degli iperparametri

4.6 Esperimento #6

L'ultimo esperimento condotto ha coinvolto l'utilizzo della tecnica di pruning per la compressione del modello. Il pruning è stato effettuato solamente sullo strato denso (circa 29 milioni di parametri) utilizzando una sparsità dell'80%. I risultati finali ottenuti sul test set sono pari al 95% a seguito del tuning degli iperparametri (condotti nell'esperimento 5) e dell'88% a seguito del pruning del medesimo modello.

5 Discussione dei risultati

I risultati ottenuti sono in linea con la valutazione teorica delle modifiche applicate, in particolare è possibile osservare come la regolarizzazione ha agito sui pesi presenti nella rete, riducendo l'overfitting e portando ad avere un trade-off accettabile tra accuratezza raggiunta sul training set e quella sul validation set che ha portato ad avere un modello più robusto in grado di generalizzare meglio il problema. Durante gli esperimenti è stato controllato lo stato di utilizzo delle risorse dell'ambiente utilizzato; sono emerse le seguenti criticità:

- il cono di bottiglia durante il training è dettato dal processore: il tempo medio di training per ogni epoca utilizzando la CPU è stato di 40 minuti (2400 secondi) contro 96 secondi utilizzando la GPU messa a disposizione da Google Colab
- l'utilizzo della RAM non ha determinato problematiche: mediamente sono stati utilizzati 4 GB su 12 GB disponibili

Possibili implementazioni future riguardano l'utilizzo di tecniche più elaborate di compressione del modello ed un utilizzo di tecniche di *"distributed tuning"* per effettuare un training efficiente della rete con un dominio degli iperparametri più ampio.

6 Conclusioni

Le tecniche di regolarizzazione applicate (data augmentation, dropout, early stopping e weight decay) si sono rilevate estremamente utili per rendere il modello robusto; i risultati ottenuti permettono di affermare che l'utilizzo del

transfer learning per il passaggio di conoscenza è efficace (97% di accuratezza sul validation set contro il 22% non utilizzando i pesi pre-allenati) e che è possibile comprimere il modello mantenendo un ottimo trade-off tra dimensione dello stesso ed accuratezza sul test set: 80% di sparsità dello strato denso con un'accuratezza dell'88% (rispetto ad un'accuratezza del 95% ottenuta con il modello non compresso).

Riferimenti bibliografici

[1] (2020) Transfer learning & fine-tuning. [Online]. Available: https://keras.io/guides/transfer_learning/

[2] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, "What is the state of neural network pruning?" *arXiv preprint arXiv:2003.03033*, 2020.

[3] (2021) Vast.ai. [Online]. Available: https://vast.ai/

[4] M.-E. Nilsback and A. Zisserman, "Automated flower classification over a large number of classes," in *Indian Conference on Computer Vision, Graphics and Image Processing*, Dec 2008.

[5] I. Guyon, "A scaling law for the validation-set training-set size ratio," *AT&T Bell Laboratories*, vol. 1, no. 11, 1997.

[6] (2020) Keras metrics. [Online]. Available: https://keras.io/api/metrics/

[7] (2020) Optimize machine learning models. [Online]. Available: https://www.tensorflow.org/model_optimization

[8] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[9] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 1251–1258.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015. [Online]. Available: http://arxiv.org/abs/1512.03385