Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Applicazione di reti neurali e *fine tuning* sul dataset *Caltech 256*

Autori:
Lorenzo Doneda - 806727 - l.doneda@campus.unimib.it
Luca Maggi - 866654 - l.maggi@campus.unimib.it
Daniele Rizzo - 872359 - d.rizzo20@campus.unimib.it

21 febbraio 2022

Indice

Sommario

Il dataset reso disponibile dalla Caltech rappresenta una sfida per la costruzione di un classificatore a causa del numero molto elevato di classi presenti al suo interno. Questo paper vuole accettare tale sfida proponendo diverse soluzioni per lo sviluppo di un modello ottimale da utilizzare in contesti similari. Per la natura dei dati a disposizione sono utilizzate prevalentemente reti neurali convoluzionali. Dopo aver implementato un modello da zero, constatandone i limiti, è proposto un approccio di *transfer learning* utilizzando come base due differenti reti già consolidate, una VGG16 e una DenseNet121. Partendo da queste sono ricavate e confrontate differenti soluzioni e, alla fine, è proposto il modello migliore. Questo è basato su una DenseNet121, tagliata per recuperare features di alto livello, e con esso è stato possibile ricavare una *accuracy* del 75% e una *top 5 accuracy* del 90%.

1 Introduzione

L'università americana Caltech (California Institute of Technology) ha messo a disposizione il dataset noto con il nome *Caltech256*[1]; esso contiene più di 30 000 immagini divise in 257 classi distinte. Queste immagini, riguardanti gli oggetti più svariati, proprio per la loro diversità e per il numero elevato di classi in cui sono categorizzate rappresentano un'interessante sfida per la costruzione di un modello predittivo automatico. Partendo da queste premesse, il paper in oggetto ha come scopo proprio quello di implementare una procedura di classificazione per il dataset; in particolare, l'obiettivo è raggiunto utilizzando delle reti neurali convoluzionali. Dopo una fase di *pre-processing* dei dati, due approcci principali sono confrontati: lo sviluppo di un modello nuovo, partendo da zero, e l'utilizzo di alcuni modelli già pre-addestrati, in un'ottica di *fine tuning*. I risultati sono successivamente commentati comparati sul due metriche: *accuracy*, intesa come top 1, e *top 5 accuracy*. Alla fine è presentato il modello migliore ottenuto insieme ad una sua disamina e una proiezione dei suoi punti di forza e di debolezza.

2 Dataset

Il dataset utilizzato è conosciuto come *"Caltech 256"*, questo nome evocativo è costituito dall'unione dell'istituzione da cui è messo a disposizione, il *California Institute of Technology*, e dalle 256 classi principali in cui le 30 607 immagini contenute al suo interno sono suddivise. A queste si aggiunge una classe extra, detta *clutter*, di rumore, portando il totale effettivo delle classi a 257. Una prima analisi del contenuto rivela che le immagini sono molto differenti tra loro, variando da esempi di animali a oggetti tra i più disparati, e nessun fattore comune può essere rintracciato. Per un utilizzo ottimale della memoria a disposizione le immagini sono suddivise per l'analisi in 234 batches da 128 esemplari l'uno, ottenuti con un campionamento casuale. Il dataset totale è quindi scomposto in tre subset, uno per il training del modello, contente 180 batches (77% del totale); uno per la validazione del modello in fase di fitting, con 30 batches (13% del totale) e, infine, uno per il test del modello su dati non utilizzati nel training, contenente 24 batches (10% del totale). Nel decidere i criteri per la divisione è stata posta particolare enfasi sulla grandezza del training set, anche a discapito di un test set relativamente minuto. Questo a causa della natura delle reti neurali

che, avendo una enorme quantità di paramenti da addestrare, necessitano di training set i più ampi possibile. Il pre-processing delle immagini è svolto con due metodologie differenti, una per i modelli costruiti da zero e una per i modelli pre-addestrati. In questo ultimo caso le immagini sono state preprocessate in modo automatico, importando il pacchetto con le stesse modalità con le quali sono state trattate le immagini originali su cui le reti neurali sono già state addestrate. Questa scelta deriva dalla volontà di preservare una omogeneità tra le immagini nuove e quelle a cui la rete è già abituata, nella speranza di poter impiegare in modo ottimale le features già presenti nel modello importato. Per quanto riguarda la costruzione delle reti create da zero viene applicato un *resizing* per avere la certezza che tutte le immagini siano della medesima grandezza di 256 x 256 x 3. Data la sullodata necessità di una grande quantità di dati per l'addestramento ottimale di una rete neurale alcune operazioni di *data augmentation* sono compiute. In particolare il dataset è aumentato attraverso l'aggiunta di immagini sottoposte ad un cambiamento casuale del contrasto dei colori, una rotazione casuale limitata al 20% dei pixel e un *rescaling* dei valori per portarli da un range da 0 a 255 ad un range da 0 a 1. Tutte queste operazioni sono compiute tenendo a mente la natura del dataset, infatti le tecniche ed i limiti scelti sono funzionali a non creare immagini senza senso che possono ingannare il modello in fase di training. Lavorando con immagini di oggetti reali non avrebbe senso ribaltare completamente un'immagine o tagliarla così drasticamente da snaturarla completamente. Infine, come ultima nota, dall'analisi della distribuzione delle immagini nelle rispettive classi salta all'occhio un notevole sbilanciamento. Alcune classi presentano una quantità di immagini molto maggiore delle altre, e, in generale, le immagini per ogni classe sono abbastanza altalenanti in numero. Per evitare di dover prendere un subset di immagini per ogni categoria in numero uguale alle immagini disponibili per la classe meno numerosa, contente solo 30 esemplari, un calcolo di pesi specifici è effettuato. Questa strategia consente di utilizzare tutte le immagini, con il beneficio che questo comporta in fase di training, ma, al contempo, evita che il modello si specializzi troppo nel riconoscere le categorie di immagini più numerose tralasciando le classi con meno esemplari. Questo è possibile grazie ad una attribuzione di pesi per ogni singola categoria di valore inversamente proporzionale al numero di campioni nella categoria stessa.

3 Approccio metodologico

3.1 Rete neurale convoluzionale

3.1.1 Architettura

Come introdotto si seguono due diverse tecniche di sviluppo. Il primo riguarda lo sviluppo di una rete neurale convoluzionale da addestrare per intero sul nostro dataset. La struttura del modello è composta da quattro blocchi convoluzionali, ognuno dei quali composto da un layer di convoluzione, un'operazione di batch *normalization*, un'attivazione ed infine un'operazione di *max pooling*. L'obiettivo è quello di utilizzare una rete non particolarmente profonda, aspetto che potrebbe consentire di evitare un problema di *vanishing gradient*, e di ottenere comunque delle valide rappresentazioni sia di alto che basso livello. I layer convoluzionali sono tutti composti da un kernel di dimensione 3×3 e nei quali si sceglie di utilizzare un'operazione di *zero padding* per conservare la dimensione di input anche in output, così da ottenere una riduzione della dimensione esclusivamente all'operazione di *pooling*. Con i layer più profondi aumenta il numero di filtri utilizzati, i valori utilizzati nei quattro layer convoluzionali, in ordine di profondità verso l'output, sono: 16, 32, 64 e 128. La scelta di questi numeri risiede nel fatto che si vuole mantenere una buona rappresentazione nonostante la riduzione della dimensione della rappresentazione per via dell'operazione di *max pooling*.
Successivamente a questo layer di convoluzione è inserito un layer di batch *normalization* che consente, attraverso la normalizzazione dell'input per il layer successivo, di stabilizzare la fase di apprendimento e ridurre il numero di epoche necessarie per un valido adattamento sui dati di train.
Una volta utilizzata questa operazione di normalizzazione si utilizza una funzione di attivazione. Per i primi layer, caratterizzati da un minore numero di filtri si utilizza la funzione "selu". Questa deriva dalla funzione ReLU ma con la particolarità di non annullarsi per argomenti minori o uguali a zero. Questa caratteristica consente di evitare il problema dei *dying neurons* che potrebbe essere particolarmente scomodo con pochi parametri a disposizione. Nei due layer più profondi invece si utilizza la funzione ReLU siccome il problema di un eccessivo numero di neuroni spenti viene superato dal numero di neuroni utilizzati. Come già sottolineato in precedenza, al termine di ogni blocco convoluzionale si effettua un'operazione di max pooling con kernel di dimensione 2×2 in modo da dimezzare la dimensione in output.

Esclusivamente per i due layer convoluzionali più profondi viene inoltre considerato un *dropout* con probabilità $p = 0.3$ in modo da contrastare il fenomeno di *overfitting* nel quale si può incappare con modelli caratterizzati da elevata capacità.
Al termine di questi blocchi di convoluzioni si effettua un'operazione di *flattening* in modo da rompere l'informazione spaziale ed elaborare la rappresentazione ottenuta con dei *layer fully connected* e svolgere la classificazione. Il primo layer è composto da 256 neuroni, un *dropout* con $p = 0.2$ e funzione di attivazione ReLU. Elaborato l'input con questi neuroni viene effettuata nuovamente una normalizzazione prima di arrivare al layer di output con 257 neuroni e funzione di attivazione *softmax*.

3.1.2 Apprendimento

Per svolgere il training del modello si utilizza come algoritmo di ottimizzazione *Adam* [2] in modo da sfruttare sia un *learning rate* adattivo, con valore di partenza $lr = 0.001$ e decadimento inversamente proporzionale al numero di epoche utilizzate, che la stabilità del momento del gradiente. La funzione scelta per questa classificazione *multi class* è la *categorical crossentropy* la quale si accoppia con la funzione softmax scelta come attivazione nel layer di output.
Dato il tipo di problema che si sta affrontando, per la valutazione delle prestazioni del modello e del suo corretto apprendimento, si monitora, oltre al valore di *accuracy* e di *loss*, il valore di *top-5 accuracy*, una misura indicata per questi casi data la presenza di ben 257 possibili classi. Una precisazione importante riguarda il peso delle singole osservazioni sul training del modello. Data la presenza di classi anche molto sbilanciate, per far si che il modello non sia a sua volta sbilanciato verso alcune particolari classi, si utilizzano dei pesi in modo che ogni classe sia ugualmente trattata dal modello.

3.2 Transfer learning

3.2.1 DenseNet

Il secondo approccio che si utilizza è il *transfer learning* in modo da sfruttare un modello pre-addestrato e specializzarlo sul caso in particolare. Il modello di base è DenseNet121 [3] addestrato con il dataset Imagenet. Data l'elevata similarità tra le immagini contenute in Imagenet e quelle del dataset Cal-

tech256, è possibile utilizzare features di alto livello prodotte dal modello di riferimento. Si sviluppano due diversi modelli con *fine tuning*, nel primo si considerano bloccati tutti i layer pre-addestrati e si utilizzano due *layer fully connected* per adattare il modello sul dataset di lavoro, mentre nel secondo si sceglie di addestrare nuovamente anche l'ultimo dei blocchi convoluzionali della rete DenseNet121 in modo da ricercare un migliore adattamento del modello sui dati. Questo tipo di architettura è particolarmente profondo e utilizza i *residual blocks* per affrontare il problema del *vanishing gradient*.

Un passaggio fondamentale per l'utilizzo di *transfer learning* è il preprocessing il quale deve essere applicato sul dataset come fatto sul dataset Imagenet in fase di addestramento del modello DenseNet121. Altro aspetto delicato riguarda la dimensione dell'input. Come descritto precedentemente le immagini si sono scelte ridimensionate in una dimensione $256 \times 256 \times 3$ ma, siccome il modello DenseNet121 è stato ideato per una dimensione $227 \times 227 \times 3$ è necessario utilizzare questa dimensione per il dataset di lavoro in modo da non incorrere in un errato utilizzo del modello.

Per il primo modello sviluppato si effettua la classificazione delle immagini arricchendo il modello di base con due layer *fully connected* rispettivamente composti da 2048 e 512 neuroni prima di un layer di output come descritto per il modello interamente costruito. Data l'elevata capacità fornita al modello da questi tre layer si impone anche un *dropout* sul primo di questi con probabilità pari a $p = 0.2$. Su tutti questi, ad eccetto del layer di output, viene utilizzata la funzione ReLU come per l'attivazione.

Il secondo modello viene invece lasciato libero di apprendere anche l'ultimo blocco convoluzionale e si utilizzano sempre due layer *fully connected* prima dell'output come in precedenza. La scelta di effettuare fine tuning è dovuta alla volontà di verificare l'impatto che un maggior numero di parametri possa avere nell'adattamento del modello sui dati.

Per entrambi i modelli vengono mantenute le condizioni di addestramento viste per la rete neurale convoluzionale del paragrafo 2.1.2.

3.2.2 VGG16

Oltre all'utilizzo di DenseNet121 si verificano le capacità dell'architettura VGG16 [4]. Questo modello utilizza una diversa logica, difatti per evitare il problema del *vanishing gradient* non sfrutta i *residual block* ma concatena

più layer convoluzionali per ottenere features di alto livello con una rete non particolarmente profonda. Il numero di parametri e di complessità risulta superiore e richiede un tempo maggiore per l'apprendimento. In questo caso si sviluppa un solo modello utilizzando tutti i layer convoluzionali e si specializza sul caso particolare utilizzando due layer *fully connected* composti da 2048 e 512 neuroni. Data l'elevata capacità del primo layer viene impostato un *dropout* con probabilità $p = 0.2$ in modo da contrastare l'*overfitting*. La funzione di attivazione utilizzata è anche in questo caso ReLU. In output è presente anche in questo caso un layer composto da 257 neuroni con funzione di attivazione *softmax*.

4 Risultati e Valutazione

Le metriche che si utilizzano per la valutazione delle performance dei modelli sono la top 1 accuracy e la top 5 accuracy, ognuna di queste metriche fa riferimento ai dati di training, di *validation* e di test. Successivamente alle metriche di prestazione vengono inserite per ciascuno dei modelli le immagini riguardanti il comportamento dell'*accuracy top 1* e della *loss*, sia per il training sia per il *validation*, così da poter analizzare l'andamento delle curve durante le diverse epoche ed effettuare un confronto.

4.1 Rete neurale convoluzionale

I valori ottenuti dalla rete neurale convoluzionale in termini di *accuracy* sono i seguenti:

Training		Validation		Test	
Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
0.53	0.76	0.37	0.58	0.32	0.53

Tabella 1: Valori di accuracy rete convoluzionale

L'andamento della fase di apprendimento del modello è il seguente:

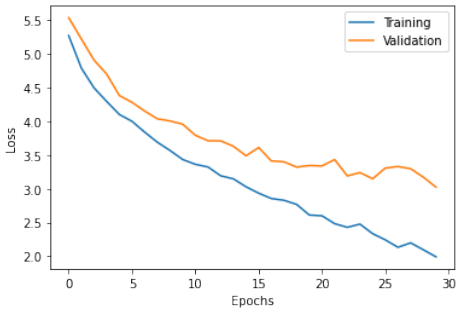

Figura 1: Loss rete neurale

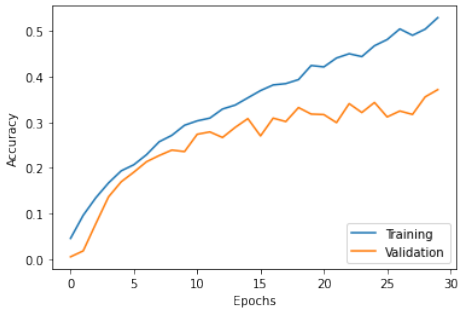

Figura 2: Accuracy rete neurale

4.2 Transfer learning

Per la parte di *transfer learning* vengono introdotti i risultati del primo modello sempre in termini di *accuracy*.

Training		Validation		Test	
Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
0.66	0.87	0.74	0.90	0.73	0.90

Tabella 2: Valori di accuracy primo modello

Il comportamento della funzione di *loss* e dell'*accuracy* durante le diverse epoche è il seguente:

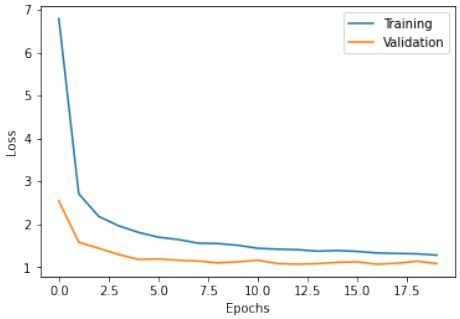

Figura 3: Loss primo modello

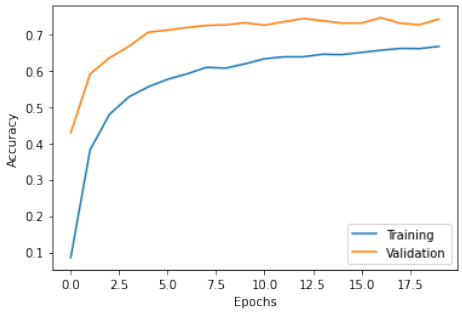

Figura 4: Accuracy primo modello

Le performance di *accuracy* che si ottengono per il secondo modello di *transfer learning* sia per il per il training sia per il test sono:

Training		Validation		Test	
Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
0.76	0.92	0.74	0.90	0.75	0.90

Tabella 3: Valori di accuracy secondo modello

Successivamente vengono presentate le curve di *loss* e di *accuracy* riguardanti il secondo modello di *transfer learning*.

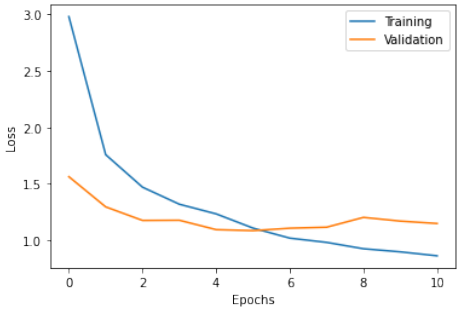

Figura 5: Loss secondo modello

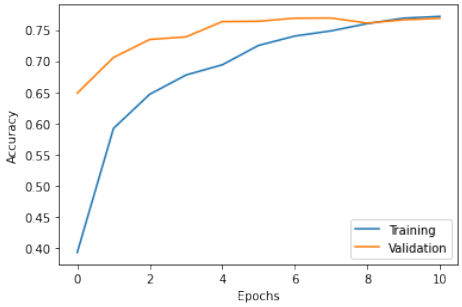

Figura 6: Accuracy secondo modello

Infine vengono presentati i risultati ottenuti per il modello VGG16:

Training		Validation		Test	
Top 1	Top 5	Top 1	Top 5	Top 1	Top 5
0.60	0.82	0.66	0.85	0.65	0.84

Tabella 4: Valori di accuracy VGG16

Le due curve di *loss* e di *accuracy* per il modello VGG16 sono le seguenti:

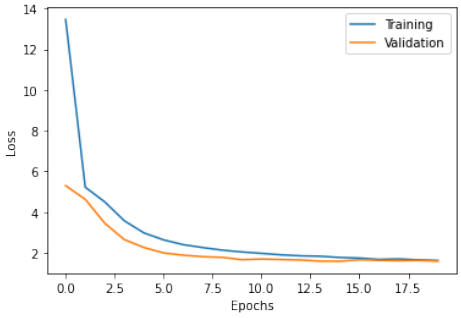

Figura 7: Loss VGG16

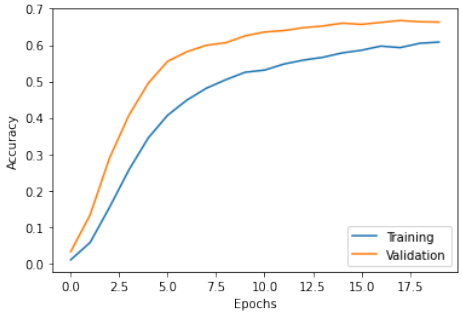

Figura 8: Accuracy VGG16

5 Discussione

Tutti i modelli sviluppati sono stati in grado di fornire dei risultati soddisfacenti.

La rete neurale convoluzionale sviluppata a partire ha mostrato una certa difficoltà nell'apprendere la natura dei dati. Le metriche analizzate risultano essere contenute ed evidenziano anche un leggero *overfitting*. Data la complessità del dataset e le risorse a disposizione si può considerare comunque un valido punto di partenza per lo sviluppo di un modello più performante. Come apprezzabile anche dall'evoluzione nel corso delle epoche il modello non presenta *underfitting* e una capacità sufficiente per adattarsi sui dati di training.
Si sono rivelati sicuramente più efficaci i modelli sviluppati tramite *transfer learning*. Il primo modello basato su DenseNet121 e specializzato esclusivamente con l'utilizzo di layer *fully connected* ha fornito dei risultati decisamente superiori di quanto ottenuto con il primo dei modelli sviluppati e inoltre non si evidenzia alcun problema di *overfitting*. Sia il valore di *accuracy top1* che *top-5* sono superiori per i set di test e *validation*, il che potrebbe indicare la necessità di approfondire il processo o di incrementare la flessibilità del modello. Questo risultato è stato ricercato con il secondo dei modelli basato su DenseNet121. Per questo difatti si è scelto di non bloccare l'apprendimen-

to dei parametri per un intero blocco convoluzionale predefinito all'interno dell'architettura. Questa scelta permette di aumentare il numero di parametri stimabili e di conseguenza di incrementare l'adattabilità del modello sui dati di training. L'operazione sembra essere eseguita con successo. Le prestazioni per gli insieme di test e validazione risultano essere analoghe a quanto visto in precedenza ma i risultati sui dati di training sono decisamente migliori ed equilibrati. Non si notano problemi di *overfitting* o *underfitting*. Le metriche mostrano inoltre che il modello risulta essere particolarmente efficace. Il valore di *top1 accuracy* è sicuramente limitato, ma data la presenza di 257 classi risulta essere comunque soddisfacente. Si rilevano dei risultati decisamente migliori considerando invece il valore di *accuracy top-5*, la quale evidenzia come il modello sia comunque decisamente in grado di interpretare i dati forniti. Nel dettaglio l'apprendimento questo modello viene fermato utilizzando una logica di *early stopping* che consente di conservare un ottimo equilibrio tra le prestazioni sui diversi set di dati.
L'ultimo modello sviluppato è basato sull'architettura VGG16. Con questo si vogliono comparare le prestazioni e capacità offerte da diverse architetture su uno stesso problema. Come per il primo modello basato su DenseNet si considerano bloccati tutti i parametri dei layer *convoluzionali* e si addestra la rete esclusivamente su layer *fully connected*. Le prestazioni risultano essere decisamente superiori alla rete neurale ideata e mostrano un comportamento del modello analogo a DenseNet in condizioni di apprendimento similari, con valori di *accuracy* superiori per i set di validazione e test. Anche questo modello potrebbe dunque trarre beneficio da una maggiore capacità.

6 Conclusioni

L'obiettivo di classificare accuratamente le immagini contenute nel dataset di riferimento in generale è stato raggiunto. Come previsto lo sviluppo di un nuovo modello partendo da zero si è rivelata una strategia non applicabile a causa del basso numero di dati a disposizione rispetto ai parametri da addestrare. Per questo motivo la soluzione ottimale è quella di performare un *transfer learning*. Delle architetture prese in considerazione, una VGG16 e una DenseNet121, quest'ultima ha ottenuto risultati migliori. A causa della similarità tra il dataset di riferimento utilizzato nell'addestramento dei sullodati modelli, Imagenet, e quello della Caltech è risultata opportuna una estrazione di *feature* di alto livello. Questo ha comportato un taglio per

entrambi i modelli molto vicino al loro output. In particolare, dopo vari tentativi, la soluzione migliore si è rivelata quella di utilizzare una DenseNet121 il cui ultimo blocco convoluzionale è stato riaddestrato e a cui sono stati aggiunti due layer *fully connected* personalizzati. Rispetto agli altri modelli ottenuti con procedura di *transfer learning* in questo modo si ottiene la giusta flessibilità per una adeguata classificazione, ciò è evidente anche osservando le performance di addestramento del *training* e *validation set* durante le varie epoche. Il modello in esame, oltre a ottenere un risultato migliore e più equilibrato, è anche quello che riesce ad essere addestrato in meno epoche, 10 in tutto. In generale le limitazioni riscontrate hanno a che vedere con la natura particolarmente sbilanciata del dataset e la variabilità delle classi presenti. La prima è affrontata con una pesatura differente per ogni classe in fase di apprendimento, proporzionale al numero di immagini per ogni categoria. Questo, per quanto efficace, non è una procedura perfetta e sicuramente, avendo a disposizione un dataset più bilanciato, i risultati potrebbero essere migliorati. In secondo luogo, sebbene la variabilità delle classi può aiutare il modello che, al contrario, potrebbe confondersi quando i dati presentati sono troppo simili per poterne ricavare delle *feature* significative, il numero di categorie nel dataset di riferimento ha rappresentato un problema. Questo è facilmente verificabile paragonando la *top 1 accuracy* con la *top 5 accuracy*, generalmente il modello ha performato molto bene nella seconda, ma meno brillantemente nella prima metrica. Come nota a margine anche le limitate risorse computazionali hanno giocato un ruolo nell'addestramento del modello. Oltre al tempo mediamente elevato per un training adeguato, più di una volta la procedura ha subito delle interruzioni a causa delle limitate risorse di calcolo a disposizione. In generale si può comunque concludere che con il giusto approccio di *transfer learning* è possibile costruire un classificatore che riesca a gestire anche un dataset con un numero di classi sufficientemente elevato.

Riferimenti bibliografici

[1] *Caltech256* http://www.vision.caltech.edu/Image_Datasets/Caltech256/

[2] *Adam: A Method for Stochastic Optimization* Diederik P. Kingma, Jimmy Ba, Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015

[3] *Densely Connected Convolutional Networks* Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, CVPR 2017

[4] *Very Deep Convolutional Networks for Large-Scale Image Recognition* Karen Simonyan, Andrew Zisserman, Published as a conference paper at ICLR 2015