Reti convoluzionali, CIFAR-10 e complessità

Riccardo Belli

***Abstract*—Il task di Image Classification ci consente di poter classificare delle immagini in base al loro contenuto visivo, estraendone quindi un significato semantico. Il dataset che andremo ad utilizzare è il CIFAR-10, composto da 60000 immagini RGB con dimensioni 32x32 pixel. Verranno esplorati due approcci: il primo con una rete convoluzionale custom e il secondo con il transfer learning, usando una rete di riferimento nella letteratura come feature extractor. Con i risultati si mostrerà come anche un modello di rete convoluzionale molto semplice può portare a dei risultati soddisfacenti, e anche quanto sia potente da un punto di vista rappresentativo avere più personalizzabilità possibile. Verranno infine evidenziati i limiti che presenta un approccio simile e le strade da poter intraprendere per ottenere dei miglioramenti nei risultati.**

I. Introduction

Nel risolvere un task di Image Classification il compito è quello di dover attribuire ad una immagine una delle etichette tra quelle disponibili. Questo significa che, a differenza di molti altri compiti classici della computer vision, ci basta solamente questa informazione; non dovremo dire dove si trova l'oggetto nell'immagine con una bounding box o definire le classi di appartenenza pixel per pixel. Un'immagine, una classe tra un set di classi predefinite, senza sovrapposizioni.

In questo progetto è stato impiegato il dataset CIFAR-10, contenente 60000 immagini a colori di dimensioni 32x32 pixel[1]. Una delle peculiarità di questo dataset è il fatto che visivamente, le classi sono completamente mutualmente esclusive; non ci sarà quindi nella stessa immagine un'auto e un camion, due delle categorie di immagini contenute nel dataset. Si tratta di un dataset molto diffuso e ampiamente utilizzato per scopi di benchmarking di algoritmi di visione, creato come subset del dataset *80 Million Tiny Images*[2]. Le classi presenti sono: *airplanes, cars, birds, cats, deer, dogs, frogs, horses, ships*, e *trucks*; inoltre è anche artificialmente bilanciato, con 6000 immagini per classe. Non avremo perciò problemi legati a sbilanciamenti durante la classificazione.

Per tutti i motivi appena elencati, è un dataset che viene ampiamente utilizzato come benchmark di algoritmi di classificazione; la sua struttura regolare consente un'ottima ripetibilità degli esperimenti e di confronto di metriche simili attraverso molteplici architetture.

Nel nostro caso, per affrontare il task di classificazione, andremo ad utilizzare due differenti architetture di reti convoluzionali. Vedremo un primo approccio che farà uso di *transfer learning*, andando ad utilizzare come estrattore di feature la rete convoluzionale ResNet-18 preaddestrata, e una seconda architettura di rete convoluzionale custom realizzata *ad hoc* per il problema, anche se non ci sono delle linee guida unificate per la definizione di architetture di questo tipo, ma solo delle *best practices*.

Vedremo infine come le risorse computazionali a disposizione possano diventare un collo di bottiglia molto velocemente in applicazioni di questo genere.

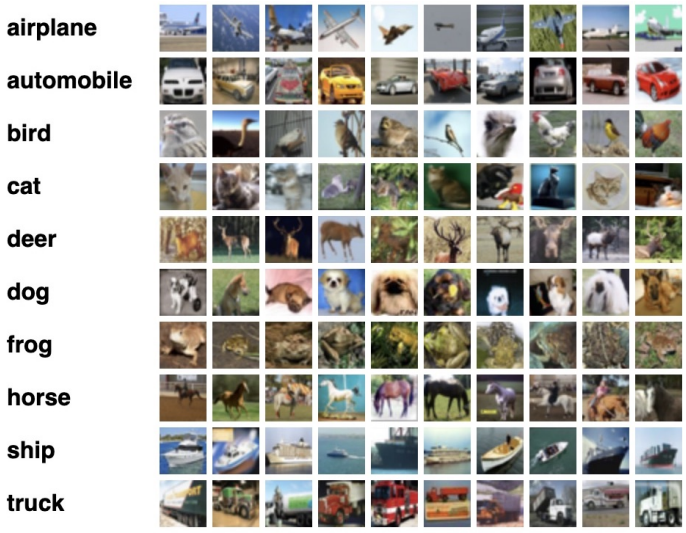

Fig. 1: Sample di immagini prese dal dataset CIFAR-10 e le relative classi di appartenenza.

II. Related Work

Esiste moltissima letteratura nell'ambito dell'Image Classification tramite reti convoluzionali; una veloce ricerca su motori di ricerca come Google Scholar restituisce decine di migliaia di risultati tra pubblicazioni e libri[3]. Alcune delle architetture più consolidate nell'ambito della Computer Vision per effettuare task di classificazione di immagini sono VGGNet[4], AlexNet[5], GoogLeNet[6] e ResNet[7]. Quest'ultima è la rete che andremo ad utilizzare per il nostro esperimento, ed è anche l'architettura che ha dato inizio alla cosiddetta "revolution of depth" delle reti convoluzionali. Anche se la versione che andremo ad usare è composta da 18 layer, esistono versioni di ResNet che hanno più di mille layer[8].

Una delle pubblicazioni più recenti, del 2019[9], riporta un'accuratezza fino al 99% per la classificazione di immagini di CIFAR-10; si può affermare quindi che si tratta di un dataset molto testato.

III. Proposed Approach

La struttura del dataset CIFAR-10 prevede 50000 immagini di training e 10000 immagini di test; anch'esse sono bilanciate tra le varie classi. Abbiamo quindi a disposizione 5000 immagini di training e 1000 immagini di test per ogni classe.

Primo approccio: transfer learning, uso di ResNet-18 come estrattore di features.

Il nostro primo approccio sarà quello di utilizzare una strategia di *transfer learning*; utilizzeremo infatti una rete convoluzionale preaddestrata dalla libreria di PyTorch[10] e la riutilizzeremo come estrattrice di features. Non dovremo quindi allenare da zero la rete per trovare tutti i pesi da assegnare ai numerosi layer convoluzionali. Grazie a librerie come PyTorch questa operazione è molto semplice: possiamo importare direttamente la rete e rimpiazzare l'ultimo layer fully connected con un layer denso avente 10 nodi in uscita, tanti quanti il numero di classi da discriminare. Si può poi impostare come fissi i pesi dei neuroni dei layer convoluzionali, lasciando all'ottimizzatore solo il lavoro sull'ultimo layer fully connected.

Una delle peculiarità di ResNet che la rende particolarmente performante è quella dell'introduzione del **residual block**, vedi Fig. 2.

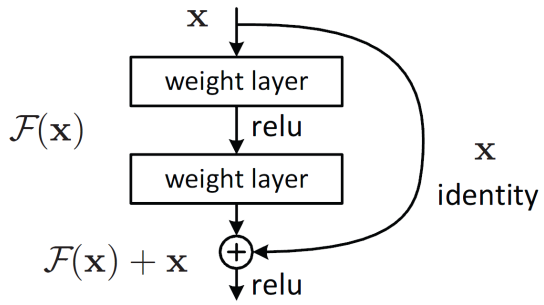

Fig. 2: Immagine presa dal paper originale di ResNet: [7]

Questo consiste semplicemente nell'inoltrare l'attivazione di un certo layer ad uno strato più in profondità nella rete; come possiamo vedere nella Fig. 2, il risultato dell'attivazione proveniente da un layer precedente viene sommata a quella di uno strato più profondo. Questo semplice accorgimento consente di evitare problemi numerici di gradiente in fase di ottimizzazione, e previene la perdita di informazione durante l'addestramento di reti molto profonde.

Vedremo in seguito che nonostante questo metodo dia la possibilità di poter riutilizzare della conoscenza pregressa senza dover ogni volta partire da zero con l'addestramento di una rete convoluzionale, non sempre il transfer learning è un procedimento efficiente per risolvere task di image classification.

Secondo approccio: rete convoluzionale custom.

Come secondo metodo, utilizzeremo una rete convoluzionale creata appositamente per il task, e addestrata sul dataset partendo da zero. I suoi layer sono definiti come segue:

- **INPUT**
- **CONVOLUTIONAL 1**
  - Numero filtri: 32
  - Dimensione Kernel: 3x3
  - Padding: SAME
  - Stride: 1px
- **RELU**
- **MAX POOLING 2D**
- **CONVOLUTIONAL 2**
  - Numero filtri: 64
  - Dimensione Kernel: 3x3
  - Padding: SAME
  - Stride: 1px
- **RELU**
- **MAX POOLING 2D**
- **CONVOLUTIONAL 3**
  - Numero filtri: 128
  - Dimensione Kernel: 3x3
  - Padding: SAME
  - Stride: 1px
- **RELU**
- **FULLY CONNECTED 1**
  - Nodi Input: 8192 $(8 * 8 * 128)$
  - Nodi output: 512
- **BATCH NORMALIZATION**
- **RELU**
- **DROPOUT**
  - $p = 0.5$
- **FULLY CONNECTED 2**
  - Nodi Input: 512
  - Nodi output: 10
- **DROPOUT**
  - $p = 0.5$

La funzione di loss utilizzata per tutti gli addestramenti è la Cross Entropy Loss, anche chiamata Log Loss e, nel caso generico multiclasse con n classi, è formulata come segue:

$$L = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p_i) \tag{1}$$

dove con p è indicata la probabilità *softmax* per la classe i, e y_i è la *truth label*. Non è assolutamente l'unica funzione di loss disponibile; ce ne sono altre come la *hinge loss*, o *logistic loss* e altre ancora, ma questa funzione è sicuramente una delle più usate; come affermato in *Pattern Recognition and Machine Learning*[11]:

"[...] using the cross-entropy error function instead of the sum-of-squares for a classification problem leads to faster training as well as improved generalization."

Dopo i primi due layer convoluzionali abbiamo utilizzato dei **MAX POOLING** per ridurre la risoluzione dei filtri e rendere la rete invariante a traslazioni dell'input, prendendo solamente il massimo di uno dei valori della regione di appartenenza. Questo aumenta anche il *receptive field* dei neuroni più in profondità, consentendo alla rete di gestire più informazioni semantiche alla volta nei filtri successivi, creando feature map via via più complesse.

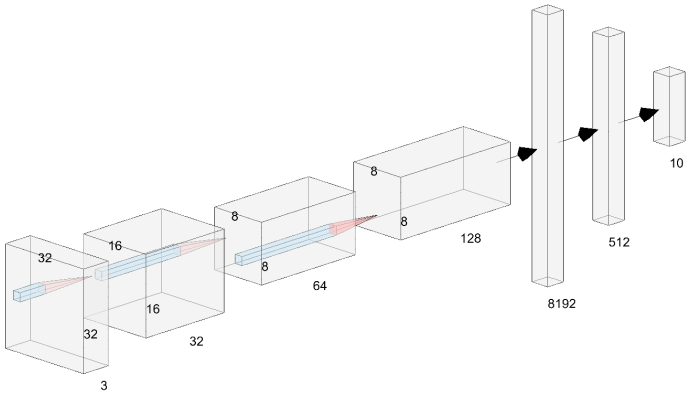

Fig. 3: Visualizzazione isometrica dell'architettura *custom net* appena descritta.

Sono stati inseriti dei **DROPOUT** dopo entrambi degli ultimi due layer fully connected, appena dopo la non linearità; questo è stato l'approccio originale degli autori che hanno proposto questo tipo di layer[12]. Da una delle ricerche più recenti si mostrano dei vantaggi ottenibili nell'utilizzare questo tipo di layer anche dopo le prime convoluzioni, sempre dopo le non linearità, anche se con dei valori di p molto più bassi, attorno a $0.1 - 0.2$[13].

Lo scopo di questo tipo di layer è quello di regolarizzare la rete, forzandola ad adattarsi a connessioni interrotte in maniera casuale con probabilità p ad ogni passo di forward (e quindi solo durante l'addestramento). Questo fa sì che ad ogni passo di training il layer su cui viene applicato il dropout viene visto e trattato come se avesse un numero di nodi e una configurazione differenti rispetto al passo precedente. Il risultato finale è quello di approssimare, o meglio, simulare, l'addestramento di più reti con differenti architetture, in parallelo.

Inoltre, è stato inserito un layer **BATCH NORMALIZATION** dopo il primo strato fully connected per aiutare la convergenza e la stabilità della rete, attraverso il ri-centramento e il re-scaling dei layer di input[14]. Era ritenuto che questa pratica riducesse lo shift della covarianza interna dei parametri, un problema legato all'inizializzazione della rete, ma degli studi più recenti mostrano come il motivo dell'aumento di di performance non è dovuto a questo effetto di normalizzazione[15]. In una pubblicazione recente, viene mostrato che usare una tecnica di *clipping* del gradiente e tramite alcuni accorgimenti sul tuning di determinati iperparametri viene resa marginale la necessità di una batch normalization[16].

IV. Experiments

Sono stati eseguiti quattro *training* differenti:

- 1) ResNet-18:
  - Ottimizzatore: Adam e SGD; scelto il modello migliore.
  - Learning rate schedulata da 1×10^{-3} a 8×10^{-5}
  - Loss: Cross Entropy
- 2) Custom net 1:
  - Ottimizzatore: Adam
  - Learning rate fissa a 1×10^{-3}
  - Loss: Cross Entropy
- 3) Custom net 2:
  - Ottimizzatore: Adam
  - Learning rate schedulata da 1×10^{-3} a 8×10^{-5}
  - Loss: Cross Entropy

Tutte le reti sono state addestrate per un totale di 30 epoche, e questo per un duplice motivo: se da un lato il tempo di addestramento iniziava a diventare quasi proibitivo per un numero di epoche maggiore, dall'altro si è visto come anche all'aumentare delle epoche (e della diminuzione della learning rate nei casi 1) e 3)) non si ricavava nessun beneficio sulla metrica di test accuracy. Potrebbe essere stata anche implementata una tecnica di *early stopping*, ma si è ritenuto utile lasciar completare lo stesso il training per tutte e 30 le epoche programmate, per verificare comunque di non trovarsi in una situazione particolare. Inoltre, si sarebbero potuti riscontrare risultati migliori grazie alla continua diminuzione della learning rate.

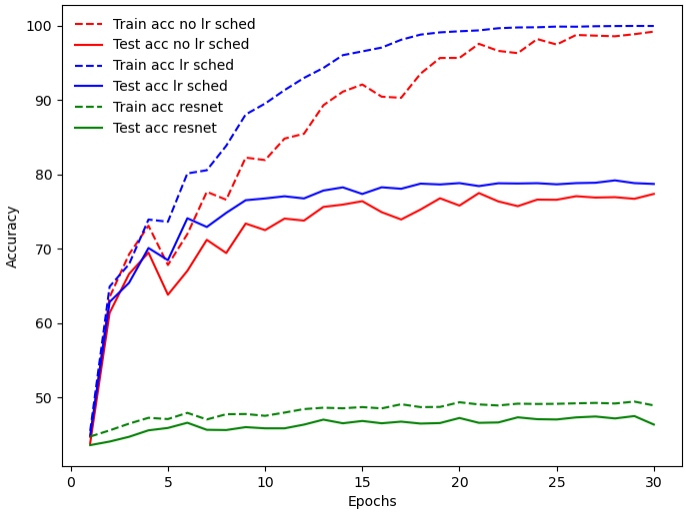

Fig. 4: Grafico delle accuratezze sullo split test/train per le architetture appena discusse.

Risultati rete custom

Analizziamo quindi i risultati di questo grafico: come possiamo ben vedere, dopo circa 10-15 epoche l'accuratezza di test è già quella definitiva, sia per il classificatore con learning rate scheduler che per quello senza, e si attesta attorno ad un 79% per la rete con scheduler contro un 77% per la rete senza. La differenza è piccola, ma rimane comunque rilevabile per tutte le epoche di addestramento della rete, dimostrando quindi l'utilità di questo approccio all'ottimizzazione con progressiva diminuzione della learning rate.

Un'altra cosa di cui tenere conto è che l'accuratezza sul test set in entrambi i casi arriva fino al 99%, quindi completamente in overfitting(?) per quanto riguarda il test set.

Risultati transfer learning con ResNet-18

Per quanto riguarda i risultati ottenuti con la seconda

architettura, e cioè quella che usa ResNet-18 come estrattore di features, i risultati purtroppo lasciano molto a desiderare. Nonostante anche in questo caso si sia fatto uso di una learning rate in diminuzione, i risultati di accuratezza raggiungono quasi subito il loro massimo: l'accuratezza sul test set sfiora al massimo il 48%, e quella sul train set raggiunge un valore più stabile attorno al 50%. Provando ad investigare il problema, ho notato che la rete è stata pre-addestrata sul dataset ImageNet[10][17]. Quest'ultimo è composto da milioni di immagini, divise in più di 20000 categorie. La teoria che ho quindi formulato è che la relativamente poca profondità della rete potrebbe aver causato un problema di collo di bottiglia nel potere di rappresentabilità della rete stessa rispetto ad una così grande mole di informazioni, ma è solamente un'ipotesi.

Vediamo ora le confusion matrix: per la rete addestrata con architettura custom considereremo solo il modello che fa uso di learning rate progressiva, in quanto riporta dei risultati complessivamente migliori.

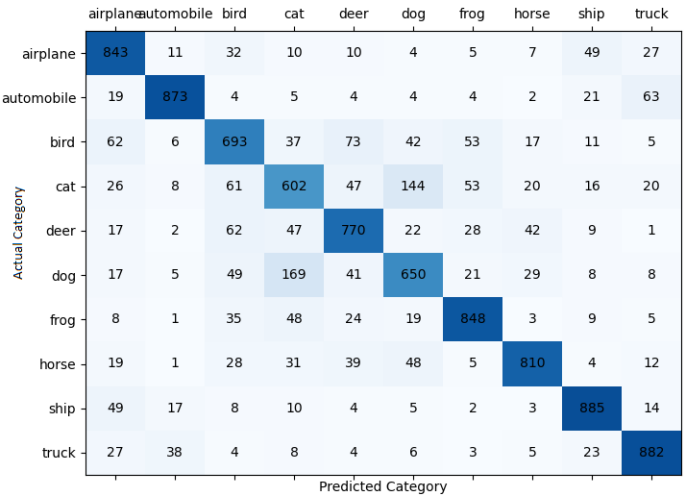

Fig. 5: Confusion matrix per il classificatore addestrato sulla rete custom.

Come vediamo dalla Fig. 5, le classi vengono identificate con una confidenza più che buona; considerata la loro bassa risoluzione, è interessante ma prevedibile vedere come le classi che vengano più confuse tra di loro siano "cat" con "dog", "airplane" con "bird" e "ship" (probabilmente a causa dello sfondo spesso prevalentemente blu), e "automobile" con "truck", anch'esse probabilmente a causa di alcune feature visive comuni tra di esse, come le ruote o gli sportelli.

Fig. 6: Due esempi di immagini presenti nel dataset non facilmente distinguibili tra di loro. Le classi di appartenenza sono: airplane, ship, dog e cat.

Vediamo come nell'esempio riportato nella Fig. 6, non tutte le immagini sono significativamente differenti tra di loro dal punto di vista visivo; alcune di esse possono facilmente essere classificate non correttamente anche da esseri umani[18]. Questo spiega i falsi positivi fuori dalla diagonale di questa matrice.

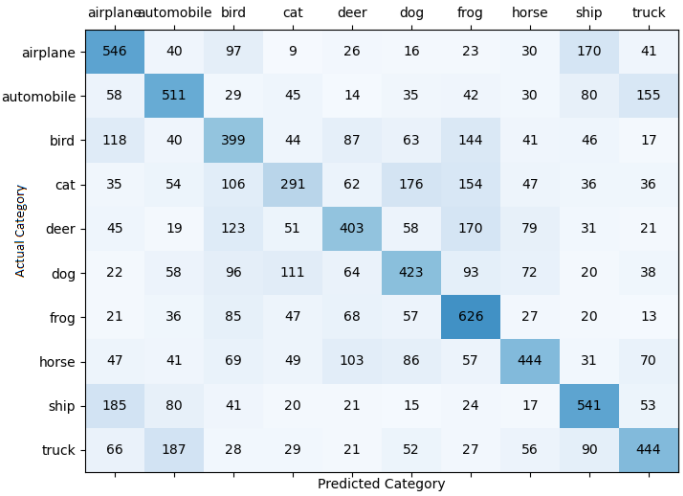

Fig. 7: Confusion matrix per il classificatore adattato da ResNet.

In questo secondo caso, come ci aspettavamo visti i risultati di Top-1 accuracy in Fig. 4, il classificatore fa molta più fatica ad identificare correttamente le classi di appartenenza delle varie immagini. Si notano le stesse confusioni discusse in precedenza ma molto più marcate, e si possono notare anche degli *outlier*: mentre ci potremmo anche aspettare che alcune immagini di "cat" a bassa risoluzione possano essere scambiate per immagini di "dog", di sicuro non ci aspettiamo che queste immagini di "cat" vengano identificate come *"frog"* con una frequenza che è la metà di quella delle classificazioni corrette.

Sulla complessità computazionale

Ritengo importante fare un accenno alla questione delle risorse di calcolo: la rete convoluzionale custom vista in questo progetto era sicuramente non in paro con la profondità e le performance di altre reti convoluzionali al vertice di competizioni come ILSVRC[19].

Come possiamo vedere dalla Fig. 8, non necessariamente un maggior numero di layer, e quindi di parametri, indica automaticamente una performance migliore, in quanto potrebbe significare solamente più problemi durante la fase di ottimizzazione. In ogni caso, al crescere della profondità della rete si può raggiungere indubbiamente una capacità rappresentativa maggiore, cosa che a me non è stato possibile sperimentare in quanto anche solo aggiungendo un ulteriore layer convoluzionale (e portandone quindi il totale a quattro) già si manifestavano i limiti fisici del computer in mio possesso, nonostante possegga una scheda grafica con 2GB di memoria video dedicata. Con la configurazione appena vista, l'utilizzo di memoria si aggira intorno ai 1.3-1.5 GB durante la fase di

training, con una *batch size* di 64 immagini. Riducendo questo iper-parametro si può ridurre l'uso della memoria, anche se, contro le mie aspettative, in maniera piuttosto marginale.

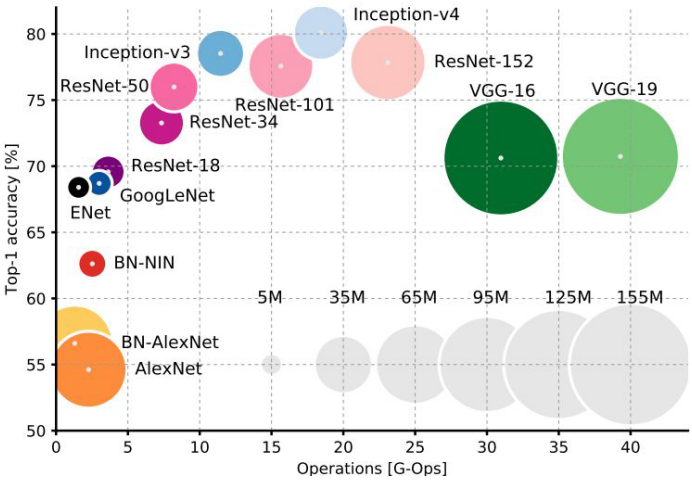

Fig. 8: Accuratezza Top-1 di alcune reti convoluzionali in relazione alla quantità di operazioni da loro richieste per il forward; la dimensione dei blob è proporzionale al numero di parametri della rete. Fonte: [20]

Sempre in relazione alla ILSVRC, possiamo vedere dalla Fig. 9 come con ResNet ci sia stata una "rivoluzione della profondità" delle reti convoluzionali: il numero di layer è esploso, e le performance di classificazione sono raddoppiate. La vera rivoluzione però, è che tutto questo non ha portato anche ad una esplosione dei parametri, e quindi delle dimensioni e complessità delle reti, ma come possiamo vedere dalla Fig. 8 il numero di parametri e le operazioni richieste per forward sono rimaste più che contenute, o addirittura diminuite in alcuni casi.

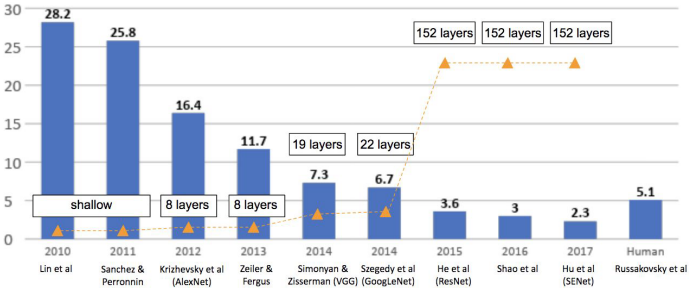

Fig. 9: Vincitori della ImageNet Large Scale Visual Recognition Challenge dal 2010 al 2017[21]. Sulle ordinate, l'errore di accuratezza Top-1.

Infine, esploriamo visivamente alcune delle *feature map* che sono state estratte dal dataset. Chiamate anche "activation map", queste indicano la risposta di attivazione di un determinato filtro al passare di un'immagine nella rete convoluzionale. Nella Fig. 10 possiamo vedere le risposte della feature map numero 6 nei vari layer, e di come viene reso sempre più marcato l'*edge* delle ali dell'aereo.

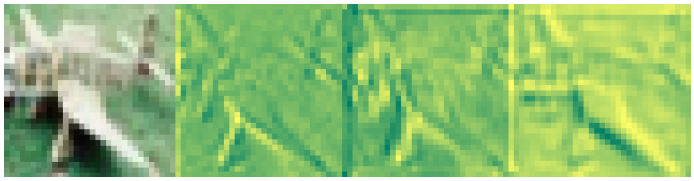

Fig. 10: Input e feature map di una immagine appartenente alla classe *airplane*.

Nella Fig. 11, possiamo vedere come l'undicesima feature map riesca a rimuovere alcuni elementi del background in quanto non informativi per l'identificazione della classe *dog*, e a mantenere una risposta elevata sui contorni della corpo del cane e sul muso.

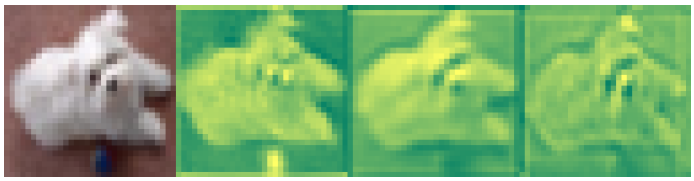

Fig. 11: Input e feature map di una immagine appartenente alla classe *dog*.

V. Conclusion

Dai risultati possiamo quindi concludere che anche una rete dall'architettura poco profonda come quella custom discussa può ottenere un'accuratezza più che soddisfacente, nonostante la sua semplice natura. Inoltre, l'aumento di complessità e le risorse di calcolo si sono rapidamente rilevate un ostacolo per la sperimentazione, non riuscendo ad andare oltre gli appena 3 layer convoluzionali.

Per quanto riguarda la rete ricavata da ResNet pre-addestrata, purtroppo le performance sono deludenti. Come discusso in precedenza, i pesi dei layer non fully connected di questa rete sono ricavati da un addestramento su un dataset differente da quello utilizzato per il resto dell'esperimento, e cioè ImageNet[10]. Solamente l'ultimo layer denso è stato riaddestrato per adattarsi al compito; tuttavia, questo approccio non si è rilevato produttivo. Nonostante ciò, questo non significa assolutamente che non sia utile la tecnica del *transfer learning*; significa invece che dovrebbe essere usata con più cura, come ad esempio con un *fine tuning* dei parametri al posto di tenerli congelati; in altri termini, partendo da un minimo, invece di rimanere lì si va verso un minimo migliore attraverso più step di ottimizzazione.

Infine, vale la pena menzionare come le classi che mettono più in difficoltà la rete sono, ovviamente, quelle più visivamente vicine, come ad esempio *cat* e *dog*, *automobile* e *truck*, e per una questione di background comune, anche *airplane* viene confuso un numero significante di volte con immagini delle classi *bird* e *ship*; un esempio è quello riportato nella Fig. 6.

Un lavoro incrementale su di questi risultati sarebbe molto semplice da realizzare concettualmente, e potrebbe comprendere l'aumentare dei layer convoluzionali, la rimozione di uno dei layer fully connected, e un migliore *fine tuning* dei parametri nel caso del riutilizzo di *transfer learning*. In pratica, questo richiederebbe delle risorse hardware aggiuntive, visto quanto discusso in precedenza.

REFERENCES

[1] Alex Krizhevsky. *The CIFAR-10 dataset*. URL: https://www.cs.toronto.edu/~kriz/cifar.html. (accessed: 07.02.2022).

[2] W. T. Freeman, R. Fergus, and A. Torralba. "80 Million Tiny Images: A Large Data Set for Nonparametric Object and Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.11 (2008), pp. 1958–1970. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2008.128.

[3] *convolutional neural network image classification - Google Scholar*. URL: https://scholar.google.com/scholar?hl=it&q=convolutional+neural+network+image+classification.

[4] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems* 25 (2012).

[6] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[7] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[8] Kaiming He et al. *Identity Mappings in Deep Residual Networks*. 2016. arXiv: 1603.05027 [cs.CV].

[9] Yanping Huang et al. *GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism*. 2019. arXiv: 1811.06965 [cs.CV].

[10] *Torchvision Models*. URL: https://pytorch.org/vision/stable/models.html#torchvision.models.resnet18.

[11] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[12] Geoffrey E. Hinton et al. *Improving neural networks by preventing co-adaptation of feature detectors*. 2012. arXiv: 1207.0580 [cs.NE].

[13] Sungheon Park and Nojun Kwak. "Analysis on the Dropout Effect in Convolutional Neural Networks". In: Mar. 2017, pp. 189–204. ISBN: 978-3-319-54183-9. DOI: 10.1007/978-3-319-54184-6_12.

[14] Sergey Ioffe and Christian Szegedy. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG].

[15] Shibani Santurkar et al. *How Does Batch Normalization Help Optimization?* 2019. arXiv: 1805.11604 [stat.ML].

[16] Andrew Brock et al. *High-Performance Large-Scale Image Recognition Without Normalization*. 2021. arXiv: 2102.06171 [cs.CV].

[17] Jia Deng et al. "Imagenet: A large-scale hierarchical image database". In: *2009 IEEE conference on computer vision and pattern recognition*. Ieee. 2009, pp. 248–255.

[18] *Dopo averle chiesto di classificare le immagini della Fig. 6, una mia amica ha erroneamente etichettato le ultime due immagini rispettivamente come "cat" e "dog", invertendo quindi le classi.*

[19] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[20] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. *An Analysis of Deep Neural Network Models for Practical Applications*. 2017. arXiv: 1605.07678 [cs.CV].

[21] Fei-Fei Li, Justin Johnson, and Serena Yeung. *Lecture 10: Recurrent Neural Networks*. 2019. URL: http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture10.pdf. (accessed: 09.02.2022).