UNIVERSITÀ DEGLI STUDI DI MILANO-BICOCCA

ADVANCED MACHINE LEARNING

PROGETTO FINALE

Classificazione di uccelli tramite reti neurali convoluzionali

---

Autori:
Matteo Cesaro - 867350 - m.cesaro1@campus.unimib.it
Francesco Martinelli - 873685 - f.martinelli21@campus.unimib.it
Cristiano Ruttico - 809360 - c.ruttico@campus.unimib.it

Gennaio 24, 2022

Indice

Abstract

L'ornitologia è una branca della zoologia che si occupa di riconoscere la classe di appartenenza degli uccelli.

La classificazione della specie è purtroppo un problema che porta con sé una grande quantità di complessità, ma che può avere vari risvolti pratici come ad esempio il più facile riconoscimento di specie a rischio.

Per affrontare questo problema si è deciso di creare un modello basato sul Deep Learning e in particolare sulle reti neurali convoluzionali.

E' stato implementato un modello attraverso la tecnica del transfer learning utilizzando ResNet50: una rete neurale già pre-trainata su un task diverso di classificazione immagini.

Con il modello creato si è potuto rispondere alla domanda di ricerca classificando la specie degli uccelli con una top-3 accuracy nel test set pari a circa l'80%.

I risultati ottenuti sono un ottimo spunto per approfondire l'argomento ulteriormente e per migliorare le performance del modello.

1 Introduzione

L'ornitologia è una branca della zoologia che si occupa di riconoscere la classe di appartenenza degli uccelli. La classificazione della specie è purtroppo un problema che porta con sé una notevole complessità sia per gli esseri umani sia che per i computer.

Sebbene diverse specie condividano una grande quantità di caratteristiche, possono spesso anche presentare molte differenze di natura macroscopica; allo stesso modo altre specie sono, a un primo impatto, quasi indistinguibili anche per i più esperti bird-watchers.

Per ovviare alla problematica di riconoscimento della classe di appartenenza si è deciso di utilizzare le reti neurali ed in particolare, siccome il dato utilizzato è costituito da immagini, le convolutional neural networks (CNN).

In particolare, si è deciso di applicare la tecnica del transfer learning, usando ResNet50 come rete dai cui estrarre le feature.

Il codice è stato scritto all'interno di Google Colab, il quale ha permesso di sfruttare le risorse computazionali messe a disposizione da Google.

2 Dataset

Il dataset per rispondere al problema di riferimento è stato ottenuto dal sito [1], in particolare il file che si ottiene dal download è denominato 'CUB_200_2011.tgz'; esso contiene 11,788 immagini di 200 specie di uccelli.

Estraendo il file tgz quello che otteniamo è una cartella chiamata 'images' al cui interno troviamo 200 sottocartelle ognuna contenente circa una sessantina di immagini di uccelli appartenente alla stessa specie; si ottengono inoltre due directory denominate 'parts' e 'attributes', le quali contengono metadati relativi alla parte del corpo dell'uccello e dei possibili attributi che quella parte può presentare; tuttavia ai fini dell'analisi in questione, queste informazioni aggiuntive non sono state utilizzate.

Le immagini presentano un'altezza e una larghezza variabile, presentano però tutte uno spazio colore di tipo RGB.

Un generico dato a nostra disposizione ha una forma del tipo: (XxYx3). All'interno della stessa cartella si presenta una varianza molto alta, dovuta principalmente alle variazioni di luce e alla molteplicità di pose che il volatile può assumere (es. uccello che vola, uccello che nuota, uccelli che sono nascosti da rami).

Alcune delle specie che possiamo riscontrare del dataset sono quelle presenti nelle seguenti immagini(Fig.1 e Fig.2).

Figura 1: Esemplare di Bohemian Maxwing

Figura 2: Esemplare di Green Kingfisher

Di seguito sono riportati due esempi di Green Kingfisher: si può notare nel primo una varianza relativa alla presenza di rami che occludono parzialmente l'immagine, mentre nel secondo una varianza dovuta alla diversa illuminazione(Fig.3 e Fig.4).

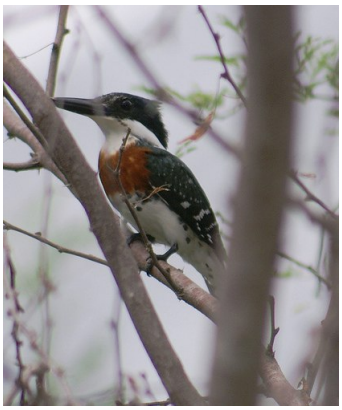

Figura 3: Green Kingfisher occluso dai rami

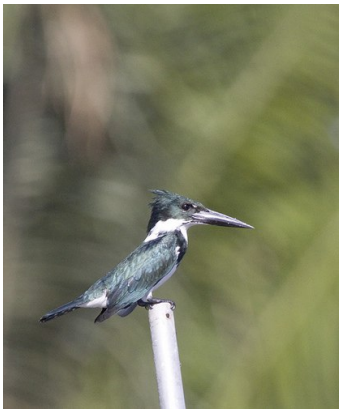

Figura 4: Green Kingfisher con diversa illuminazione

2.1 Preprocessing

Per lavorare sui dati a disposizione si sono rese necessarie alcune modifiche derivanti sia da limiti di natura computazionale, in quanto Google Colab mette a disposizione una quantità limitata di ram, la quale, se viene superata, interrompe l'esecuzione e annulla lo stato delle variabili salvate fino a quel punto, sia di natura metodologica, in quanto, prima di essere "date in pasto al modello", le immagini necessitano di essere nelle dimensioni tali che il modello possa processarle.

Dopo aver decompresso il file si nota che sebbene le immagini siano circa sessanta per ogni specie di uccello, esiste una certa variabilità nel numero di immagini nelle varie specie. Al fine di evitare distorsioni durante la fase di training, e soprattutto al fine di attenuare il problema della RAM, si è scelto di estrarre da ogni cartella un massimo di 45 immagini corrispondente circa all'80% delle immagini a nostra disposizione per ogni specie.

La fase di estrazione è stata accompagnata dalla fase di resizing dell'immagine: nello specifico si è provveduto a modificare le informazioni spaziali, imponendo la size di 224 x 224; inoltre è stato necessario applicare il metodo preprocess_input.

La modifica spaziale e il metodo preprocess_input sono stati fondamentali al fine di far si che le immagini si presentassero in una forma accettabile per ResNet50. Contestualmente si è proceduto al labelling: il processo presenta una struttura di questo tipo:

Tabella 1: Labelling

Specie	**Label**
Black_footed_Albatross	0
Laysan_Albatross	1
Sooty_Albatross	2
...	...
Common_Yellowthroat	199

Nello specifico le immagini sono state inserite all'interno della lista denominata 'ds' e le rispettive label all'interno della lista 'ds_lab'.

Entrambe le liste sono state convertite in numpy array e si è poi provveduto ad effettuare uno shuffle dei dati al fine di non introdurre bias in fase di training.

Un esempio di output ottenuto dalla fase di ridimensionamento e applicazione del metodo preprocess_input è visibile alla Fig.5.

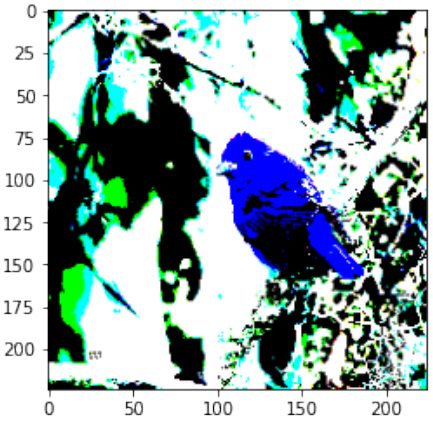

Figura 5: Immagine Pre-processata

- **Train-test split**

  Per permettere l'addestramento del modello si è proceduto a splittare i dati in

training e test: i valori scelti per lo split sono 70% per il training e 30% per il test set; contestualmente si è provveduto a impostare il random_state ai fini di riproducibilità dei risultati.

- **One-hot-encoding**

  In ultima istanza è stato effettuato un cambio di rappresentazione per quanto riguarda le label, in particolare, si è passati a una rappresentazione one-hot.

  Ottenendo delle label nella seguente forma(Fig.6):

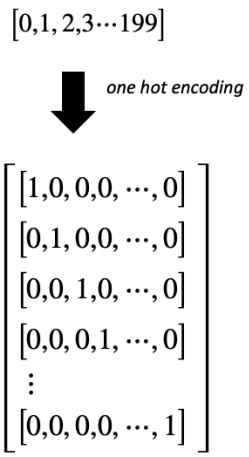

Figura 6: One hot Encoding

3 Approccio metodologico

Nel contesto della computer vision si è storicamente rivelata di notevole efficacia l'implementazione delle Reti Neurali Convoluzionali o CNN. Si è per queste ragioni deciso di utilizzare questo tipo di reti.

Una volta implementate le CNN si utilizza una Fully-Connected Neural Network come classificatore per ottenere le predizioni di classe. Un addestramento efficace di queste reti necessita tuttavia di una elevata mole di dati: nel dataset in questione il volume di immagini è inadatto a questo scopo e si presenta un serio rischio di overfitting del modello.

La soluzione intrapresa consiste nello sfruttare una rete pre-addestrata per la parte convoluzionale.

Il Transfer Learning, ovvero l'utilizzo di reti pre-addestrate, è una pratica che sfrutta architetture e pesi che si sono dimostrate efficaci su certi task per implementarli su nuovi compiti.[2]

Con questo approccio si può aggirare l'addestramento ex-novo di una rete complessa che presenta le problematiche sovra esposte. Il compito di riconoscimento degli oggetti è una sfida già affrontata con risultati ottimi nella ImageNet Large Scale Visual Recognition Challenge: la sfida consiste nel riconoscimento di 1'000 classi di oggetti in immagini su larga scala e mette a disposizione un dataset di più di 14 milioni di immagini. Si è adottata quindi una delle architetture create per ImageNet con i pesi addestrate per questo dataset.

Il transfer learning offre due possibilità: la feature extraction oppure il fine-tuning, che prevede non solo l'utilizzo di una parte della rete pre-addestrata, ma anche l'implementazione di una nuova sessione di training su tutti o alcuni dei layer già addestrati.

Solitamente questi due approcci si utilizzano in base alla dimensione del nuovo dataset e alla somiglianza del task con quello del modello originale. Nel caso trattato la dimensione del dataset risultava inadatta all'utilizzo del fine-tuning, poiché addestrare nuovamente i layer già addestrati su un dataset relativamente piccolo come quello a disposizione avrebbe incrementato il rischio di overfitting, senza considerare le tempistiche necessarie per riaddestrare adeguatamente una rete convoluzionale di dimensione analoga a quelle utilizzate nella ImageNet Challenge.

La scelta tra le varie architetture disponibili tramite la libreria di python keras si è basata sulla performance di queste sul dataset ImageNet e sulla dimensione accettata dell'input per ragioni di capacità computazionale.

Le architetture più recenti, che tendenzialmente ottengono migliori risultati, prendono come input immagini di dimensioni (299x299x3), tuttavia queste dimensioni ci consentivano di utilizzare solamente tra le 20 e le 25 delle circa 60 immagini disponibili per classe nel dataset. Nei tentativi effettuati in questo senso i risultati ottenuti erano inferiori del circa 10% in termini di accuracy rispetto ai modelli che consentivano di utilizzare più dati con risoluzione inferiore, ceteris paribus.

La scelta è ricaduta sull'architettura ResNet che ottiene risultati notevoli su ImageNet e soddisfa il requisito della InputShape (224x224x3): tra le versioni si è optato per la ResNet50 che empiricamente otteneva risultati leggermente superiori alla ResNet34 e alla ResNet152.

Nella feature extraction una decisione rilevante riguarda i layer dell'architettura originale da utilizzare. Il criterio sottostante è che più i due task sono simili più l'architettura originale è di per sé performante sul nuovo compi-

to ed è conveniente utilizzare una porzione più considerevole dell'architettura pre-addestrata.

Nel caso trattato i due compiti erano simili: la ImageNet Challenge comprende infatti il riconoscimento tra 90 razze canine e anche la classificazione tra alcuni volatili; dunque, si sono testati alcuni dei layer più prossimi all'output. Anche in questo caso un limite era rappresentato dalla RAM, in quanto appiattendo alcuni layer con output tridimensionali la dimensione della singola istanza divergeva in maniera repentina rendendone impossibile il processamento in massa.

È stata sfruttata con queste motivazioni per intero la parte convoluzionale di ResNet50 con cui si sono ottenuti risultati che superavano di almeno il 6% in accuracy tutti gli altri layer testati: il taglio è stato quindi effettuato all'ultimo layer di pooling, il global average pooling denominato "avg_pool".

Una volta estratte le feature si procede ad addestrare un classificatore fornendo le stesse come input. In primis, abbiamo tentato di rimanere coerenti con la struttura originale di ResNet50, inserendo quindi un unico Fully-Connected Layer.

Si è notata tuttavia una discreta tendenza all'overfitting, una rapida convergenza e un'incapacità di ottenere miglioramenti significativi dopo le prime iterazioni, nonostante variazioni negli iperparametri.

Questo ha portato a considerare l'aggiunta di un layer di input di 2048 neuroni che consentisse un training più equilibrato e fosse capace di incrementare la capacità discernitiva dell'unico layer.

Le funzioni di perdita e del FC Layer sono state selezionate in base alla natura del problema di classificazione: la funzione softmax per l'output layer e la categorical crossentropy come loss function rappresentano le scelte convenzionali per la classificazione multiclasse.

Gli iperparametri sono stati settati implementando la GridSearchCV, un algoritmo che sfrutta il metodo forza bruta per testare il modello compilandolo con parametri presi da delle liste fornite in input. I parametri da impostare nel classificatore erano i seguenti:

- **Learning rate**

  Il learning rate è stato inizialmente corretto analizzando i primi risultati ottenuti graficamente e poi è stato settato tramite GridSearch tra una lista di valori prossimi a quelli che empiricamente avevano offerto outcome più desiderabili. La rapidità della convergenza consente di capire se il learning rate è troppo elevato o troppo basso come mopstrato in Fig.7. Il learning rate selezionato inizialmente era troppo alto. Come learing rate è stato quindi settato 8x10-5.

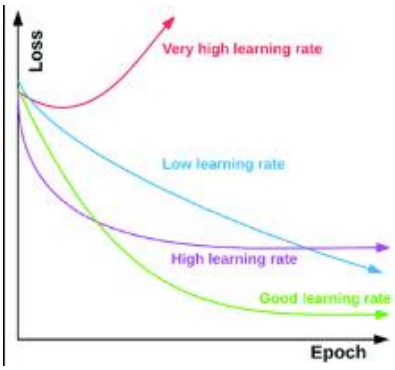

Figura 7: Learning Rate

- **Optimizer**

  Per l'ottimizzazione è stato scelto l'algoritmo Adam perché offre risultati migliori nel contesto della computer vision rispetto ad altri algoritmi di ottimizzazione classici, come lo SGD oppure Adagrad e l'RMSProp (della cui combinazione Adam è frutto) sia per efficacia che per efficienza.

- **Epochs**

  Il numero di iterazione deve tenere conto del learning rate, infatti, le epoche devono essere sufficienti a garantire la convergenza ma non così tante da favorire l'overfitting. Si è optato per 128 epoch seguendo le indicazioni della Grid Search.

- **Batch Size**

  La dimensione dei mini-batch influenza la convergenza e dipende dal learning rate utilizzato e dal numero di epochs. Il valore della batch size ottenuta dalla Grid Search con i valori precedenti per gli altri iperparametri è stata di 64.

- **Validation Split**

  Il validation split deve tenere conto della dimensione del dataset e del numero di iterazioni, infatti, se il dataset è piccolo utilizzare una porzione del training set considerevole è sconsigliato perché sono dati su cui il modello non viene addestrato in quella iterazione, al contrario un validation set troppo piccolo non offre una proxy accurata del comportamento sul test set e la

performance ottenuta nella validation tende ad avere una varianza elevata. Il valore scelto per queste ragioni è stato di 0.1.

Data la presenza di overfitting sono stati implementati alcuni metodi di regolarizzazione per ottenere prestazioni migliori sul test set. I metodi adottati sono stati:

- **Drop-out**

  E' stato aggiunto un layer di drop-out prima del dense layer. La percentuale di neuroni disattivata è un iperparametro: euristicamente si è soliti utilizzare valori nell'ordine dello 0.7 quando si tratta di layer di input e valori inferiori per hidden layer e output layer (rispettivamente 0.5 e 0.2), tuttavia nel nostro caso si trattava di un unico layer che funge da layer di input e di output per cui sono stati testati valori intermedi tra le due tipologie. Il valore che ha offerto performance migliori è stato 0.4.

- **Regolarizzazione L2**

  Un altro metodo implementato è stato quello della penalizzazione L2. Questo approccio introduce nella computazione un valore che tende a far incrementare la funzione di costo del quadrato della somma dei pesi moltiplicato per un parametro lambda, così da inibire la tendenza a una loro crescita eccessiva. Lambda è stato settato dopo alcuni tentativi sul valore di 0.05.

- **Gaussian Noise**

  Il gaussian noise è un metodo di generalizzazione che introduce del rumore nei dati. Anche in questo caso si introduce un iperparametro che regola l'intensità del rumore, più elevato è questo valore maggiore sarà la variabilità introdotta: 0.1 è un valore che ha consentito di generalizzare il modello senza perdite in termini di performance.[3]

4 Risultati e Valutazione

Il modello ottenuto è stato valutando utilizzando come misure di performance l'Accuracy e la Top k Accuracy con k = 3.

Sulla base delle misure di valutazione scelte, di seguito nella Tab.2 e nelle Fig.8,Fig.9,Fig.10 sono riportati rispettivamente i risultati ottenuti dal modello sul test set e i grafici di Loss, Accuracy e Top three Accuracy per epoca ottenuti sul validation set e il training set durante l'addestramento del modello nel suo setting finale.

Tabella 2: Risultati Test Set

	Loss	**Accuracy**	**Top-3 Acc.**
Training Set	1.543	82.09%	95.51%
Validation Set	2.421	60.56%	79.35%
Test Set	**2.545**	**61.68%**	**81.11%**

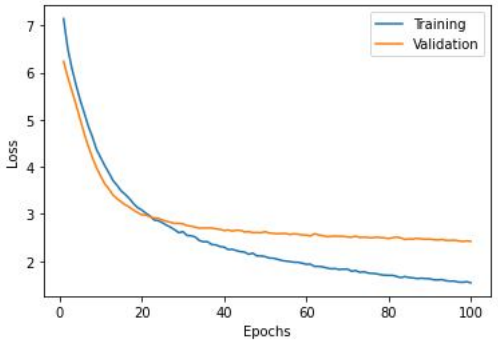

Figura 8: Loss

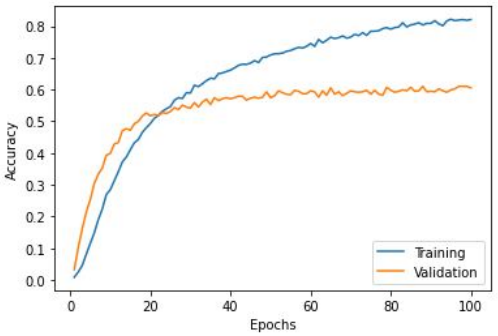

Figura 9: Accuracy

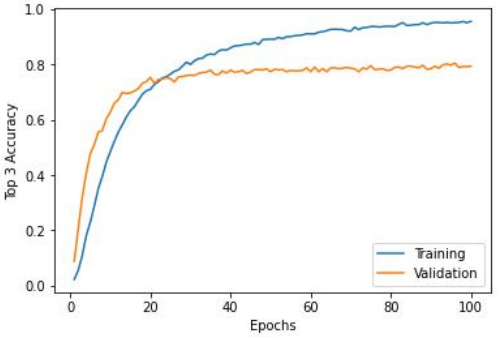

Figura 10: Top-3 Accuracy

5 Discussione

In riferimento alla domanda di ricerca posta, il valore di riferimento per la valutazione dei risultati è l'accuracy che si ottiene sul test set.

Il valore più alto ottenuto è quello del 61.67% per quanto riguarda la top one accuracy, e dell'81.11% per quanto riguarda la top three accuracy. Il modello è in grado di classificare correttamente la specie di appartenenza dell'uccello circa 6 volte su 10 per quanto riguarda la top-one accuracy, e circa 8 volte su 10 considerando la top-three accuracy.

Dalle fig.9 e fig.10 possiamo notare rispettivamente l'andamento dei valori per la top-one e top-three con lo scorrere delle epoche: in particolare si nota come l'andamento dell'accuracy fino alla ventesima epoca sia migliore per il validation set rispetto al training set, indice di un leggero underfitting.

Nella fig.8 invece notiamo l'andamento della loss rispetto al validation e al training set. Si noti come in questo caso l'andamento delle curve sia conforme a quanto atteso in teoria, al fine di ottenere i migliori risultati possibili. Alla luce di questi risultati, ci si può ritenere abbastanza soddisfatti delle performance del modello, soprattutto considerando i limiti di memoria imposti da colab che hanno impedito di mettere in atto altre strategie di regolarizzazione come la data augmentation.

Questo metodo consente la generazione di nuovi dati alterando le immagini già presenti nel dataset con delle trasformazioni di vario tipo. La condizione posta sul tipo di trasformazioni utilizzate è che queste non alterino l'outcome di classe dell'immagine originale.

Il procedimento si presta al compito, poiché ribaltamenti geometrici e rotazioni sono variazioni che il modello desiderato dovrebbe essere tendenzialmente in grado approcciare (si pensi ad esempio ai vari angoli da cui può essere fotografato uno stesso uccello in volo).

I limiti di infrastruttura non hanno inoltre permesso di sfruttare tutte le immagini a disposizione, si è presentata perciò necessità di lavorare con 9'000 immagini a fronte di una disponibilità iniziale di 11'788.

Di queste 9'000 solo 6'300 sono state dedicate al training del modello; si ritiene che nel futuro, avere a disposizione una quantità maggiore di immagini per la fase di training, possa portare ad un incremento significativo dell'accuracy.

Sempre ai fini di fornire un migliore valore dell'accuracy, si è provato a tagliare la rete pre-addestrata in un punto più vicino all'input così da intercettare una rappresentazione di più basso livello, senza però riuscire ad ottenere dei risultati migliori.

Un ulteriore miglioramento che potrebbe essere posto in atto sarebbe quello di creare un'architettura da zero, così da gestire le dimensioni dei filtri convoluzionali: in particolare la gestione diretta dei filtri che si occupano della 'local response normalization' e dello spatial pooling' fornirebbero una minore varianza all'interno della stessa classe.

Una soluzione implementabile per risolvere il problema di scala (molto presente all'interno delle immagini presenti nel dataset) è quello dell'utilizzo degli spatial transformes introdotti da Mark Jaderberg[4]: questi permettono di raggiungere l'invarianza spaziale trasformando adattivamente gli oggetti da riconoscere in input e facendogli assumere, attraverso varie trasformazioni, una sorta di posa canonica; si ritiene che questo possa condurre a migliori performance.

6 Conclusioni

Il modello sfrutta efficacemente l'architettura ResNet50 sul nuovo compito. Dei molti aggiustamenti provati, nessuno di essi ha ottenuto un miglioramento significativo delle prestazioni rispetto al valore ottenuto. Quindi, si è concluso che senza modifiche consistenti all'architettura, all'infrastruttura o al set di dati, la performance rimarrebbe intorno al livello raggiunto.

Il modello si comporta adeguatamente in particolare per quanto riguarda la top three accuracy. Per un bird watcher, essere in grado di ridurre il pool di possibili specie a tre permette di eseguire ulteriori ricerche manuali per classificare correttamente le specie. Comunque, nella maggior parte dei casi la classe è prevista correttamente.

Altre informazioni aggiuntive difficilmente ottenibili dalle foto potrebbero essere utili, per esempio la geo-localizzazione e l'ora in cui la foto è stata scattata potrebbero aiutare nella classificazione, perché l'areale di una specie è limitato e le abitudini sono utili per una corretta classificazione. Infine, potrebbe essere interessante trovare dei pattern nei risultati circa quali uccelli siano maggiormente classificati erroneamente. Con queste accortezze il modello potrebbe essere migliorato.

Riferimenti bibliografici

[1] Dataset

http://www.vision.caltech.edu/visipedia/CUB-200-2011.html

[2] Transfer Learning

https://www.tensorflow.org/tutorials/images/transfer_learning

[3] Noise

https://towardsdatascience.com/noise-its-not-always-annoying-1bd5f0f240f

[4] Spatial Transformer

https://towardsdatascience.com/spatial-transformer-networks-b743c0d112be