Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

PetFinder.my - Pawpularity Contest

Predizione Della Popolarità Di Foto Di Animali

Authors:
Francesco Lenti - 865274 - f.lenti3@campus.unimib.it
Mattia Boller - 873358 - m.boller@campus.unimib.it
Mattia Marchi - 817587 - m.marchi@campus.unimib.it

11 gennaio 2022

Sommario

Una immagine vale più di mille parole. Sapevi che un' immagine può salvare più di mille vite? Milioni di animali randagi soffrono per le strade o vengono soppressi nei rifugi ogni giorno in tutto il mondo. È chiaro aspettarsi che gli animali con foto più "attraenti" generino più interesse e vengano adottati più velocemente. Ma cosa rende "attraente" una immagine? Con l'aiuto del Deep Learning cercheremo di determinare l'attrattiva di una foto di un animale al fine di dargli una maggiore possibilità di adozione. In questo documento verranno illustrate le tecniche e le soluzioni adottate, tramite l'utilizzo di diversi modelli di deep neural network, per ottenere tale score di popolarità.

1 Introduzione

Il task fa riferimento ad una competizione aperta su Kaggle. La competizione è finanziata da PetFinder.my, principale piattaforma per il benessere degli animali della Malesia. Attualmente, PetFinder.my utilizza un misuratore per stimare la popolarità delle foto di animali randagi ospitati all'interno di rifugi. Questo misuratore analizza le statistiche riguardanti il traffico generato dalle foto degli animali, presenti su diversi portali, e ne stima la popolarità. Sebbene questo strumento sia utile, potrebbe essere ancora migliorato. L'obiettivo finale è quello di stimare a priori la "Pawpularity" (termine coniato dalla piattaforma per indicare la popolarità di una foto) di un animale in base alla foto del suo profilo, in modo di scegliere quelle che ne faciliterebbero l'adozione. Insieme alle foto di migliaia di animali vengono anche forniti dei metadati, etichettati a mano, per ogni foto. Vengono forniti quindi due dataset, uno con le foto degli animali e l'altro con i corrispettivi metadati. La soluzione da noi proposta utilizza un modello multi-input, per utilizzare tutte le informazioni a disposizione. Da un lato una rete convoluzionale che sfrutta un approccio di fine-tuning basato su transfer learning, dall'altro una semplice rete neurale fully-connected. Gli output delle due reti vengono concatenati e processati da un ulteriore layer fully-connected finale per ottenere l'output, ovvero il Pawpularity score.

2 Datasets

Vengono forniti un dataset di immagini e un dataset tabulare contenente i metadati delle immagini.
In particolare, i dati di training comprendono:

- Una cartella **train** contenente 9912 immagini di training nella forma $\{id\}.jpg$, dove id è un identificativo univoco del profilo dell'animale;
- Un file ***train.csv*** contenente i metadati e il Pawpularity score per ogni immagine.

I dati di testing non sono disponibili, dato che si tratta di una competizione Kaggle ancora aperta al momento della stesura di questo report. Tuttavia, vengono fornite la cartella **test** e il file ***test.csv*** di 8 immagini al fine di testare la submission dell'algoritmo. Inoltre è presente anche un file ***sample_submission.csv*** con un esempio di submission. Dopo aver inviato la submission, il modello verrà testato su un subset pari al 25% del dataset originale di test, contenete circa 6800 immagini, il quale verrà utilizzato solo al termine della competizione per stilare la classifica finale.

2.1 Data Exploration

Processo per investigare quelle che sono le caratteristiche chiave dei dataset. Prima di costruire un modello di machine learning è necessario capire il dataset con cui si sta lavorando e il problema che si sta cercando di risolvere. Entriamo ora nel dettaglio.

2.1.1 Immagini

Per quanto riguarda il dataset delle immagini andremo a mostrare alcune foto in base al pawpularity score assegnato. Analizziamo quindi le immagini partendo dai voti più bassi andando verso quelli più alti. Per non rendere il report troppo prolisso, sono stati inseriti solo 3 gradi di pawpularity: 1, 50 e 100, il resto dell'analisi è disponibile nel notebook.

Figura 1: Animali con Pawpularity=1

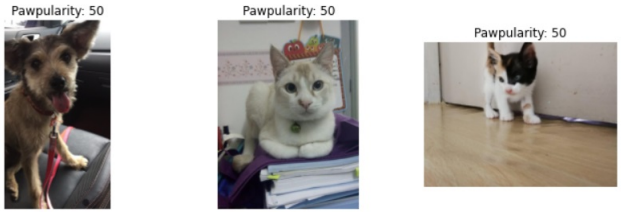

Figura 2: Animali con Pawpularity=50

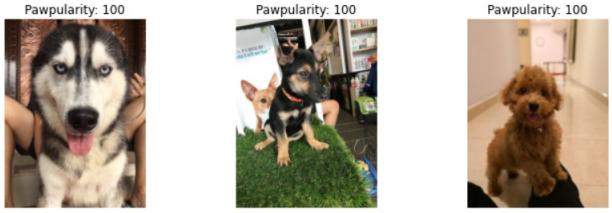

Figura 3: Animali con Pawpularity=100

Dall'analisi visiva effettuata non si evincono particolari dettagli su cosa possa rendere una foto più "popolare" rispetto ad un'altra, è facile, infatti, trovare delle belle foto di animali con score di pawpularity sotto il 10.

2.1.2 Metadati

Come descritto in precedenza il file *train.csv* contiene i metadati per ogni immagine. In figura 4 visualizziamo la distribuzione del target, ovvero il Pawpularity score.

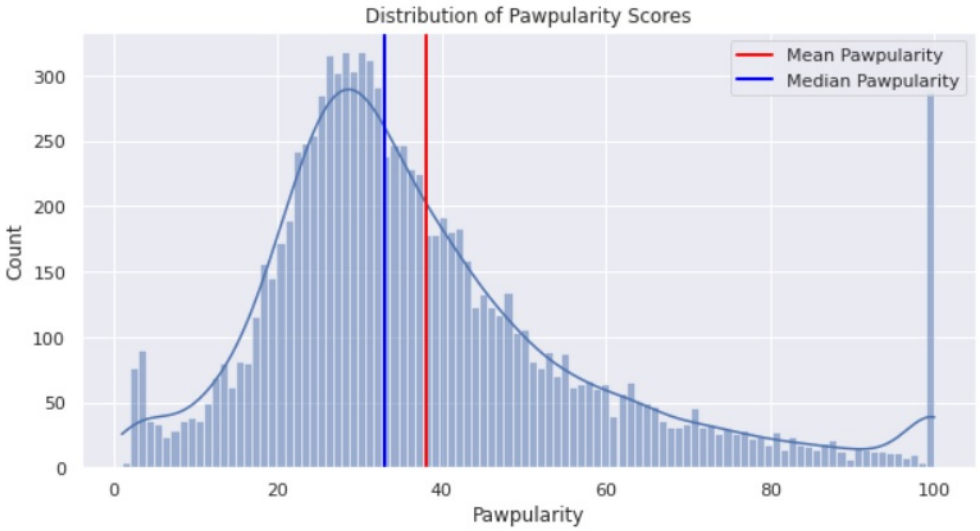

Figura 4: Distribuzione e esempio della feature Focus

Osserviamo come la distribuzione dello score sia concentrata principalmente tra i valori 20 e 50. È facile osservare anche una forte presenza di immagini con score uguale a 100. Successivamente analizziamo la distribuzione del target per ogni singola feature del dataset. Per fare ciò usiamo degli istogrammi, mostriamo il Pawpularity score sull'asse x e la somma dei valori di ogni feature sull'asse y. Questo ci può aiutare a visualizzare se alcune feature impattano più di altre nel calcolo del Pawpularity score. Per ogni feature inoltre vengono mostrati degli esempi di immagini per contestualizzare meglio il significato della feature stessa.

- **Focus** - L'animale è posizionato su uno sfondo ordinato, non troppo vicino/lontano;

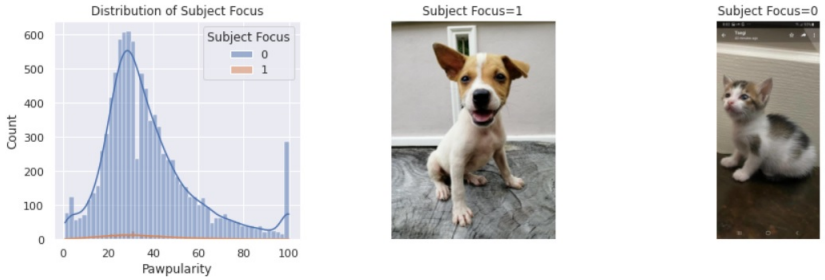

Figura 5: Distribuzione e esempio della feature Focus

- **Eyes** - Entrambi gli occhi sono rivolti verso l'obiettivo, con almeno un occhio/pupilla decentemente chiaro;

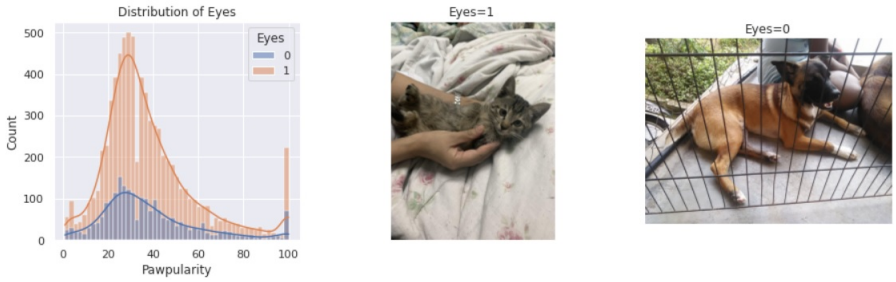

Figura 6: Distribuzione e esempio della feature Eyes

- **Face** - Viso discretamente chiaro e rivolto in avanti;

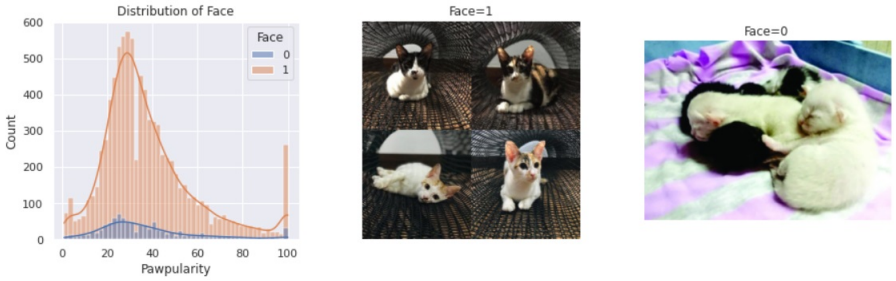

Figura 7: Distribuzione e esempio della feature Face

- **Near** - Singolo animale che occupa una porzione significativa della foto (circa oltre il 50% della larghezza o dell'altezza della foto);

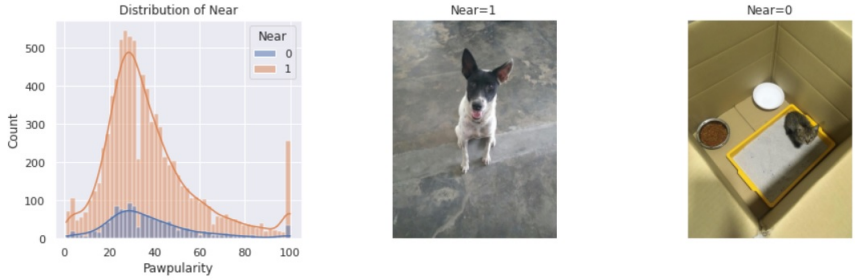

Figura 8: Distribuzione e esempio della feature Near

- **Action** - Animale nel mentre di un'azione (ad es. saltare);

Figura 9: Distribuzione e esempio della feature Action

- **Accessory** - Accessorio/oggetto di accompagnamento fisico o digitale, come un giocattolo o un adesivo digitale, esclusi collare e guinzaglio;

Figura 10: Distribuzione e esempio della feature Accessory

- **Group** - Più di 1 animale nella foto;

Figura 11: Distribuzione e esempio della feature Group

- **Collage** - Foto ritoccata digitalmente;

Figura 12: Distribuzione e esempio della feature Collage

- **Human** - Umano nella foto;

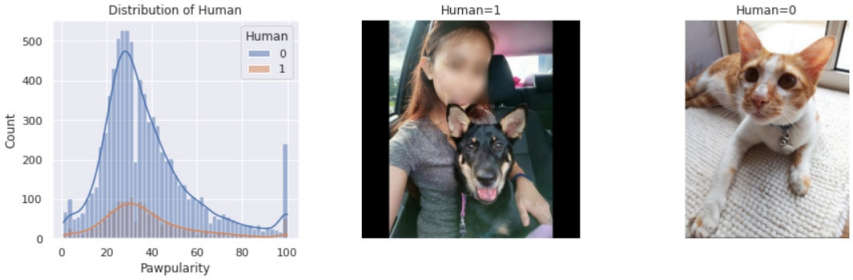

Figura 13: Distribuzione e esempio della feature Human

- **Occlusion** - Oggetti indesiderati che bloccano una parte dell'animale. Nota: Non tutti gli oggetti bloccanti sono considerati occlusione;

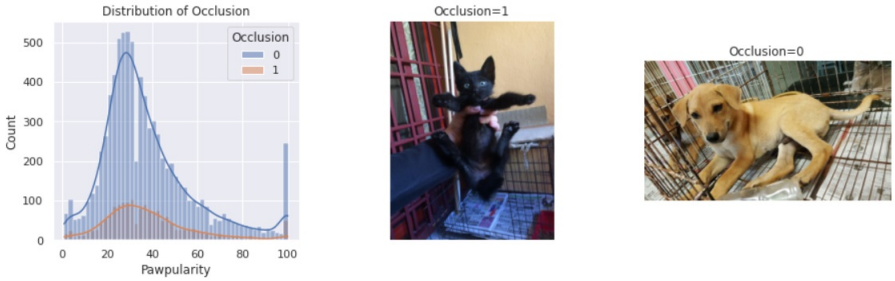

Figura 14: Distribuzione e esempio della feature Occlusion

- **Info** - Mesto o etichette personalizzati (ad es. nome dell'animale, descrizione);

Figura 15: Distribuzione e esempio della feature Info

- **Blur** - Notevolmente sfocato o rumoroso, soprattutto per gli occhi e il viso dell'animale. Per le voci Sfocatura, la colonna "Occhi" è sempre impostata su 0.

Figura 16: Distribuzione e esempio della feature Blur

Intuitivamente è possibile affermare che questi grafici non diano molte informazioni. La distribuzione del Pawpularity score è simile per ogni feature, in altre parole nessuna feature sembra influenzare particolarmente il Pawpularity score.

3 Aprroccio metodologico

La competizione proposta da PetFinder.my si presenta come un problema di regressione, con l'obiettivo di creare un modello capace di assegnare un punteggio di attrattività, compreso tra 0 e 100, ad una determinata foto di un animale. Nei seguenti paragrafi verrano presentati i vari step che hanno portato alla definizione del modello finale utilizzato per il task sopra descritto. Si è partiti da alcuni modelli di deep learning più tradizionali, quali fully-connected neural networks e convolutional neural networks, per poi cercare di combinare le informazioni contenute nelle immagini con le informazioni date dai metadati allegati alle foto. Si anticipa che l'approccio finale proposto è basato su un modello multi-input single-output. La motivazione alla base di questo approccio è quella di sviluppare un sistema di apprendimento che sia in grado di sfruttare le informazioni utili da features miste di dati, poiché questi tipi di dati di solito richiedono un trattamento separato. A tal fine, ogni tipo di dato in ingresso viene elaborato e gestito in modo diverso e indipendente.

3.1 Convolutional neural network

Il primo approccio adottato per cercare di risolvere il problema e per dare una base di partenza da cui poi definire modelli più complessi, è stato quello di sfruttare una convolutional neural network, classico tipo di rete neurale adatto a trattare immagini. Con questo primo tentativo si è cercato di capire che performance era possibile ottenere sfruttando solamente le foto, lasciando momentaneamente in disparte i metadati forniti.

Il modello costruito si basa su transfer learning e fine-tuning di una rete allenata in precedenza, in modo da poter sfruttare le elevate performance di questi modelli andando ad abbattere i tempi di allenamento da zero di una nuova rete di grandezza simile. In particolare, per il task trattato da questo caso di studio, si è optato per la rete EfficientNet-B3. Le EfficientNet sono una famiglia di reti presentate nel maggio 2019, basate su un metodo di model scaling innovativo rispetto ad esempio a MobileNet o ResNet il quale va a scalare uniformemente le dimensioni di larghezza, profondità e

risoluzione utilizzando un coefficiente composto ϕ nella seguente maniera:

$$\begin{aligned} depth &: d = \alpha^{\phi} \\ width &: w = \beta^{\phi} \\ resolution &: r = \gamma^{\phi} \\ s.t.\ &\alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ &\alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned} \tag{1}$$

dove α, β, γ sono costanti determinabili da una piccola grid search.

La versione B7 di EfficientNet ha raggiunto una top-1 accuracy su ImageNet pari al 84.3%, raggiungendo lo stato dell'arte a parità di Gpipe pur essendo 8.4 volte più piccola e 6.1 volte più veloce nell'inferenza. EfficientNet si è inoltre dimostrata molto performante in caso di utilizzo in transfer learning, in particolare raggiungendo un'accuracy pari al 91.7% su Cifar-100 e al 98.8% su Flowers. Nel caso del problema trattato in questo documento, si è scelta la versione B3, la quale prende in input immagini di dimensione 300x300, decisione dettata dal trade-off tra compattezza, quindi limiti hardware, e performance della rete. [1] Le performance delle varie versioni di EfficientNet su ImageNet rispetto ad altre reti sono mostrate in figura 17.

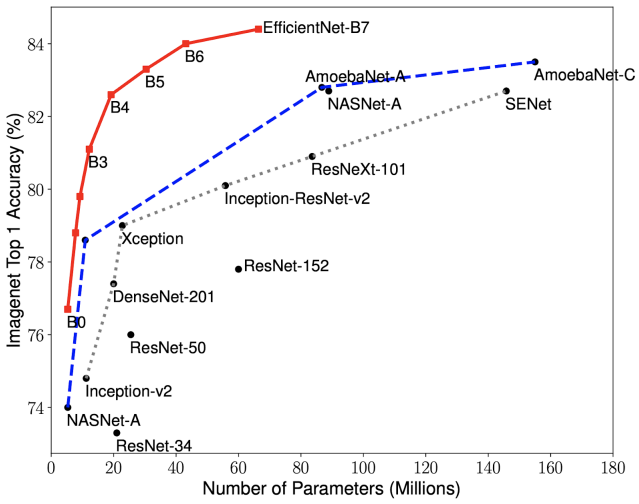

Figura 17: Performance di EfficientNet rispetto ad altre reti.

Inoltre, i pesi importati, sono quelli ottenuti tramite la tecnica di training Noisy Student. Noisy Student è una metodo di apprendimento semi-

supervisionato che si basa sull'allenare un modello (teacher) con immagini etichettate, utilizzarlo per etichettare nuove immagini e allenare un nuovo modello (student), della stessa grandezza o maggiore, con le immagini iniziali e quelle nuove ottenute dal primo modello aggiugendo del rumore. Questi passi vengono poi ripetuti più volte, invertendo teacher e student. Attraverso questa tecnica, utilizzando 300M di immagini non etichettate, EfficientNet-B7 "Noisy Student" ha raggiunto una top-1 accuracy su ImageNet pari al 88.4%. Questo approccio è stato scelto inoltre perché consente una generalizzazione migliore. [2]

In coda alla rete pre-allenata importata, sono stati aggiunti dei layer fully-connected per adattare il modello al problema di regressione di PetFinder.my. Di seguito è specificata l'architettura della sezione aggiunta:

- **Dense** - 1024 neuroni, ReLU
- **Dropout** - 50% dei neuroni eliminati.
- **Dense** - 512 neuroni, ReLU
- **Dropout** - 30% dei neuroni eliminati.
- **Dense** - 128 neuroni, ReLU
- **Dropout** - 10% dei neuroni eliminati.
- **Dense** - 1 neurone, Linear

La funzione di loss scelta è stata la Mean Squared Error, mentre per monitorare le performance è stata utilizzata come metrica la Root Mean Squared Error. Come algoritmo di ottimizzazione è stato selezionato Adam.

3.2 Fully-connected neural network

Per trattare i metadati riguardati le foto, si è deciso di utilizzare una rete neurale classica fully-connected. Di seguito viene riportata l'architettura del modello:

- **Dense** - 32 neuroni, ReLU
- **Dropout** - 20% dei neuroni eliminati.
- **Dense** - 8 neuroni, ReLU

- **Dropout** - 10% dei neuroni eliminati.
- **Dense** - 1 neurone, Linear

Anche in questo caso, come per la rete convoluzionale, si sono utilizzate MSE come funzione di loss, RMSE come metrica delle performance del modello e Adam come algoritmo di ottimizzazione.

3.3 Hybrid DNN

L'ipotesi secondo cui predizioni più accurate si possono ottenere combinando le informazioni derivanti dalle foto degli animali e dai metadati ad esse allegati, ha portato alla costruzione di un terzo modello, combinazione dei due precedentemente descritti.

Il modello in questione si presenta come una rete neurale ibrida [3], costituita dalla rete convoluzionale illustrata nel paragrafo 3.1 e dalla rete fully-connected di cui si indicano le specifiche nel paragrafo 3.2, i quali output vengono concatenati e processati da due ulteriori layer. L'architettura proposta è mostrata in figura 18.

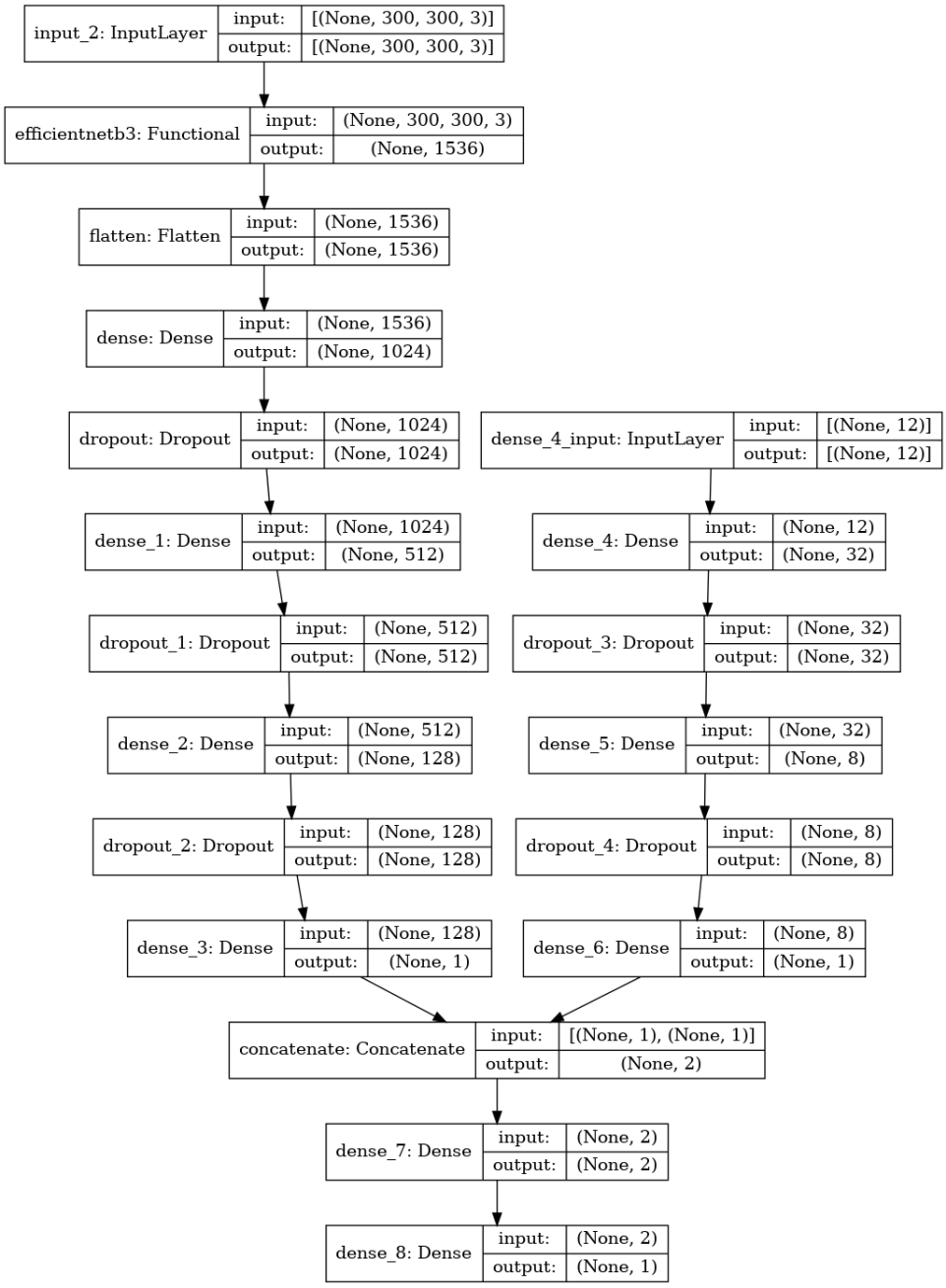

Figura 18: Architettura Hybrid DNN

3.4 Training

Il training del modello finale descritto nel paragrafo 3.3, è avvenuto allenando prima separatamente i due modelli di cui è composto. Entrambe le reti sono state allenate utilizzando una grandezza di batch di 32, per un massimo di 15 epoche, applicando in entrambi i casi early-stopping per evitare casi di overfitting.

I due modelli allenati sono stati successivamente uniti per andare a formare la rete ibrida e i loro pesi sono stati bloccati, così da poter effettuare il training solo sulla sezione aggiunta in seguito alla concatenzaione dei risultati.

Durante la fase di training del terzo modello, si è riscontrato un fenomeno di blocco in un minimo locale, il quale impediva miglioramenti delle performance. Il problema è stato affrontato effettuando uno studio sul valore ideale di learning rate, ricerca che ha portato alle scelta di un valore pari a 0.1, più alto rispetto ai valori inizialmente adottati. In figura 19 viene presentato il grafico dello studio che ha portato alla scelta del particolare valore di learning rate.

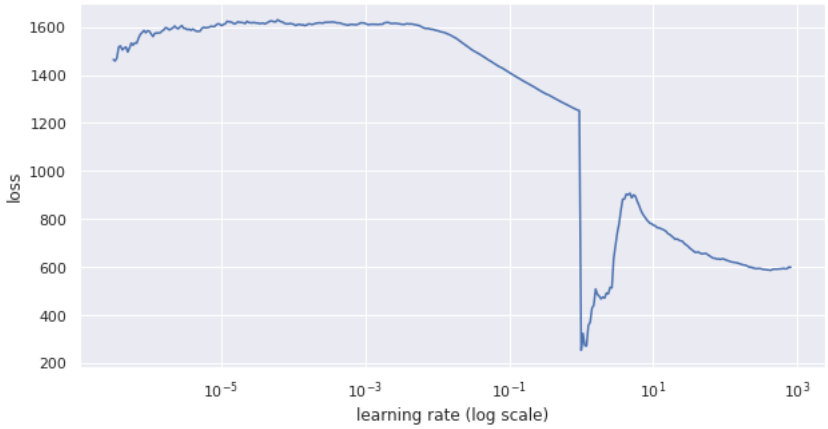

Figura 19: Grafico del valore di loss rispetto a diversi valori di learning rate

Il modello ibrido è stato allenato con grandezza di batch pari a 32, per un massimo di 30 epoche, anche in questo caso limitate da early-stopping.

4 Risultati e Valutazione

La metrica usata per la valutazione delle performance, essendo un task di regressione, è rappresentata dal calcolo del **RMSE (Root Mean Square Error)**:

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

con $\hat{y}_i$ il valore di popolarità predetto e y_i il valore originale.

Le performance vengono valutate nelle varie epoche, sia sul training set che sul validation set. Tale valutazione viene effettuata prima sui singoli modelli, figure 20 e 21, e successivamente, in figura 22, sul modello finale ottenuto tramite la concatenazione dei primi due. E' possibile notare come nella rete finale non ci sia traccia di overfitting, a differenza del modello ottenuto dalla rete convoluzionale in cui, sul train set, lo score continua a decrescere senza migliorare sul validation set.

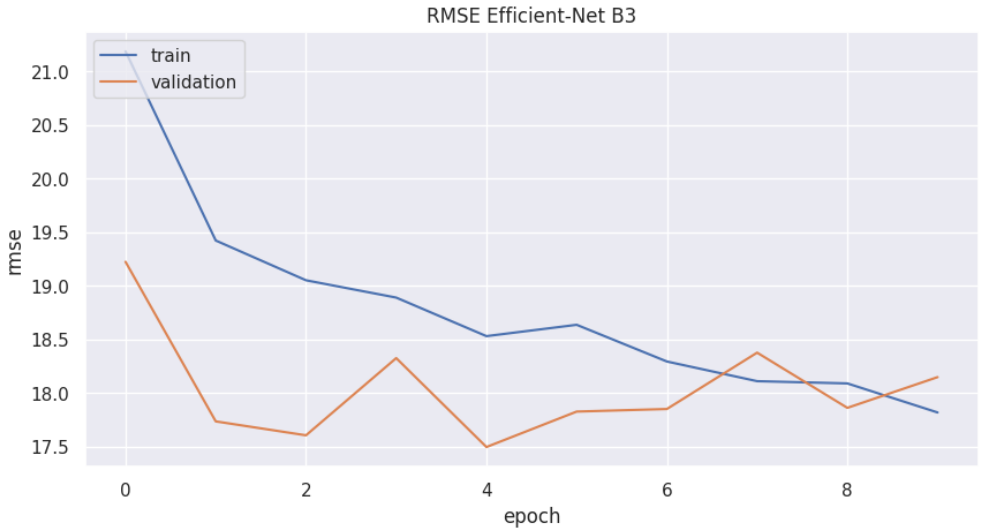

Figura 20: RMSE del modello Efficient-Net B3 su train set (blu) e validation set (arancione) durante le epoche.

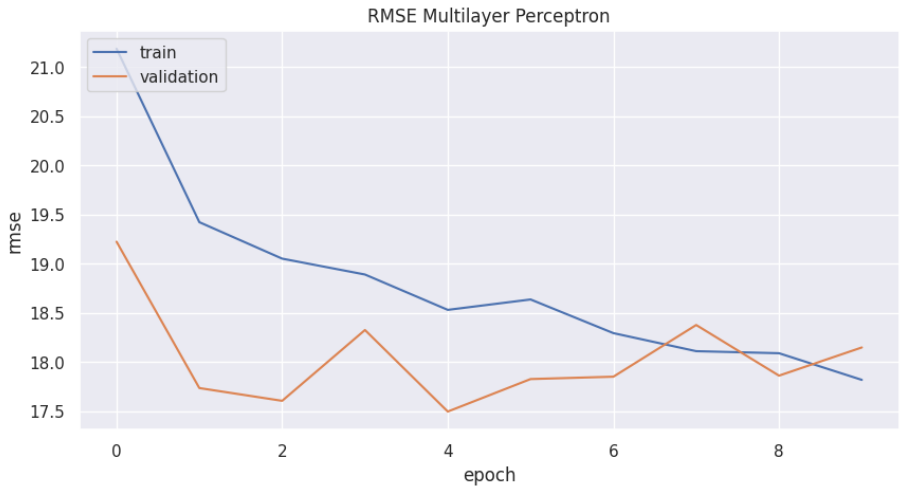

Figura 21: RMSE del modello Multilayer Perceptron su train set (blu) e validation set (arancione) durante le epoche.

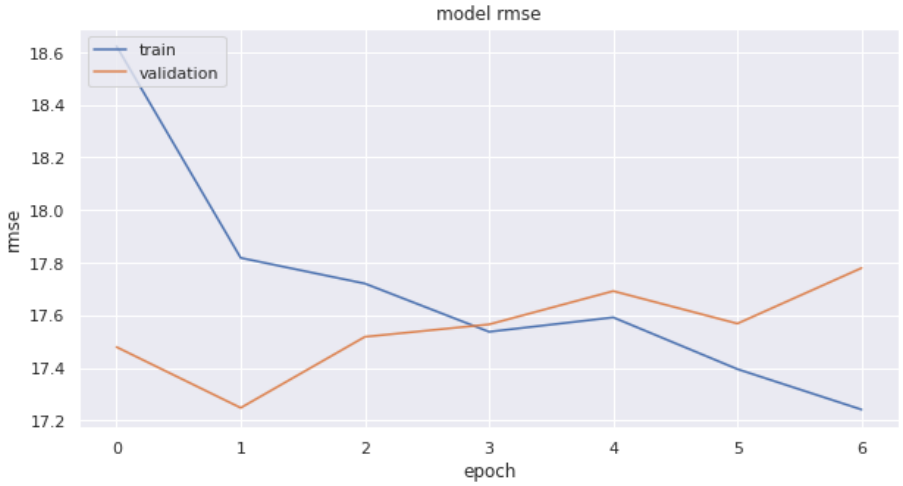

Figura 22: RMSE del modello Hybrid su train set (blu) e validation set (arancione) durante le epoche.

4.1 Confronto Risultati

Nella tabella 1 vengono raffigurati i valori migliori ottenuti sul validation set per le varie architetture. La figura 23 mostra l'andamento della metrica RMSE dei differenti modelli nelle varie epoche.

Tabella 1: Confronto RMSE ottenuta tra i diversi modelli sul validation set.

Modello	**RMSE Validation**
Efficient-Net B3	1
MLP	1
Hybrid	1

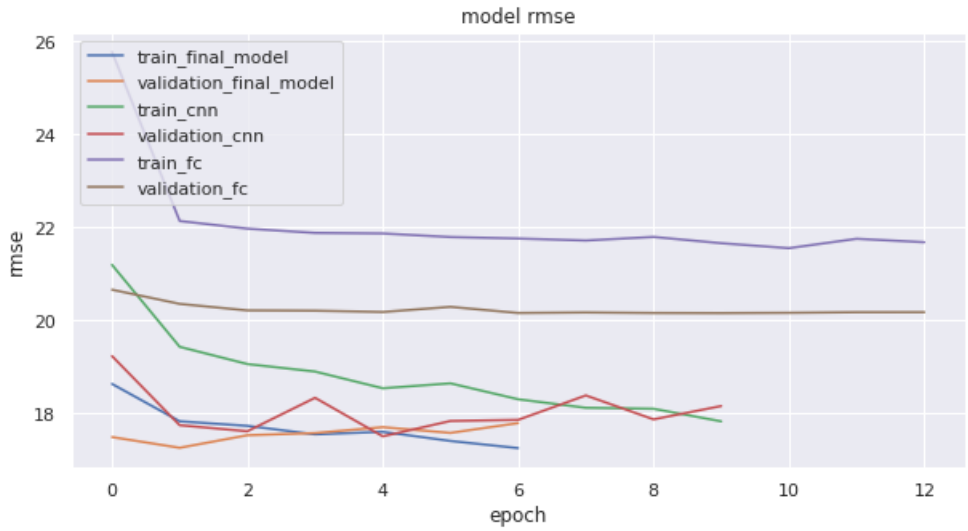

Figura 23: RMSE modelli a confronto.

4.2 Grad CAM

Un problema noto del deep learning è rappresentato dal fatto che i modelli creati siano "black box", ovvero che sia difficilmente spiegabile come realmente le informazioni siano trattate all'interno dei vari layers. Per analizzare i risultati ottenuti è stato ritenuto opportuno effettuare un ulteriore studio su come la rete convoluzionale ragionasse sulle immagini per ottenere lo score di popolarità. Questo è possibile utilizzando lo strumento **Grad CAM**[4]. Tale metodo consente di produrre una spiegazione visiva di ciò che la rete osserva, determinando quali sono gli aspetti e le features che più influenzano lo score finale. E' possibile notare nella figura 24 come la rete si vada a focalizzare maggiormente sull'animale nel caso in cui l'immagine risulti pulita da oggetti di disturbo; al contrario, se l'immagine presenta disturbi di vario genere, la rete andrà a dare maggior importanza all'ambiente circostante piuttosto che al solo animale.

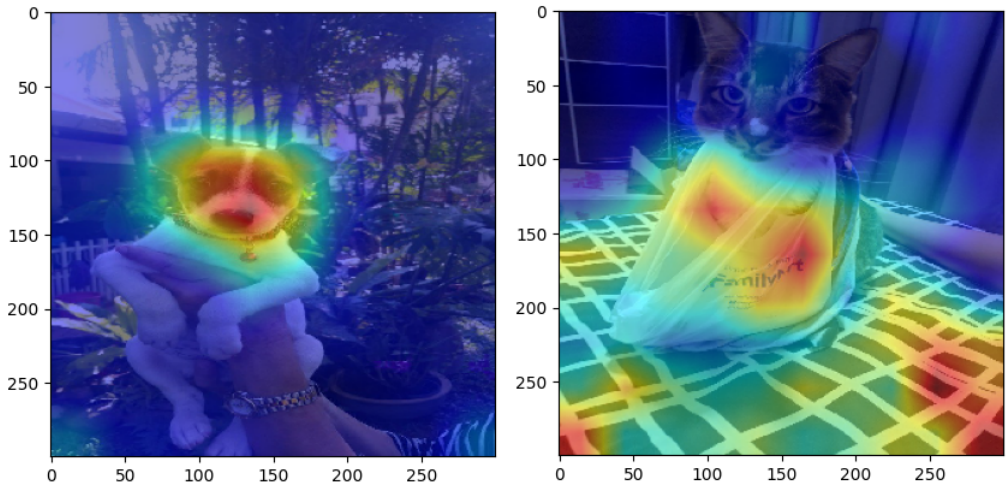

Figura 24: Confronto tra due Grad CAM ottenute da immagini con diversi score: quella di sinistra con popolarità più alta, mentre quella di destra più bassa.

5 Considerazioni

I risultati mostrano come le performance ottenute dal modello finale siano migliori di quelle ottenute tramite il solo fine-tuning sulla rete Efficient-Net B3 e nettamente migliori di quelli del Multilayer Perceptron sui meta-dati. Questo è dovuto da un utilizzo di tutte le informazioni disponibili e non solo le immagini: il layer fully-connected del modello finale riesce a dare i giusti pesi alle informazioni ottenute dalla rete convoluzionale, per le immagini, e da quella fully-connected, per i meta-dati, migliorando il risultato finale. Le performance ottenute risultano migliori di quelle raggiunte da altri team kaggle utilizzanti reti convoluzionali allo stato dell'arte. I migliori risultati tuttavia vengono ottenuti da team che sfruttano un altro tipo di modello, ovvero i **transformer**, migliorando lo score finale di un punto rispetto all'utilizzo di reti convoluzionali.

6 Conclusioni e sviluppi futuri

L'obiettivo di questo caso di studio è stato determinare se, l'uso congiunto di architetture su tipologie di dati diverse, potesse portare effettivamente a migliori risultati rispetto alle singole.

L'ipotesi formulata, sebbene non supportata da una grande quantità di materiale presente in letteratura, è stata confermata dai risultati ottenuti portando un miglioramento significativo delle performance. Tuttavia i risultati ottenuti non sono competitivi rispetto a quelli derivanti dall'utilizzo di nuove tecniche avanzate, quali transformer.

Un possibile sviluppo futuro, può essere la sostitusione della rete convoluzionale con un transformer e l'utilizzo di una risoluzione maggiore per le immagini, migliorie che devono essere necessariamente supportate da un hardware adeguato.

Riferimenti bibliografici

[1] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," 2020.

[2] Q. Xie, M.-T. Luong, E. Hovy, and Q. V. Le, "Self-training with noisy student improves imagenet classification," 2020.

[3] Z. Yuan, Y. Jiang, J. Li, and H. Huang, "Hybrid-dnns: Hybrid deep neural networks for mixed inputs," 2020.

[4] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-cam: Visual explanations from deep networks via gradient-based localization," *International Journal of Computer Vision*, vol. 128, no. 2, p. 336–359, Oct 2019. [Online]. Available: http://dx.doi.org/10.1007/s11263-019-01228-7