Anselmo Raffaele
Matricola 846842
r.anselmo@campus.unimib.it

ASSIGNMENT 4

ADVANCED MACHINE LEARNING

L'assignment consiste nella risoluzione di un problema di classificazione supervisionata di immagini attraverso il Transfer Learning. In particolare si utilizza una CNN pre-addestrata su *IMAGENET* con architettura VGG16.

1. Dataset Overview and Task Description

Il dataset scelto è il *"10-monkeys-species"* contenente 1300 immagini relative a 10 specie differenti di scimmie. Il dataset è disponibile su Kaggle: https://www.kaggle.com/slothkong/10-monkey-species

Obiettivo dell'assignment è l'utilizzo della una rete neurale convoluzionale VGG16 come feature extractor a diversi livelli per risolvere il problema di classificazione supervisionata delle immagini.

Il dataset di interesse è molto più piccolo rispetto ad IMAGENET, ma non molto differente. Si testa perciò l'ipotesi che le features estratte da VGG16 riescano ad individuare caratteristiche più dettagliate del dataset, riuscendo ad ottenere buone performances senza dover stimare un numero troppo alto di parametri.
Viene a tale scopo sviluppata una rete neurale classica che utilizzerà le features estratte a 3 diversi livelli dalla CNN VGG16: il primo utilizzerà tutti i layer ad eccezione di quello di output (Modello 1), il secondo taglierà la VGG16 al 4 blocco di layer (Modello 2) ed il terzo la taglierà 3 blocco di layer (Modello 3).

Le immagini hanno risoluzioni differenti, uguali o maggiori a 400x300 pixels, vengono perciò ridimensionate a 224x224 pixels.

Ai fini di valutazione dei modelli viene definita la funzione ***plot_metrics***.

2. Preprocessing

Per far si che le nostre immagini siano compatibili con quelle su cui è stata addestrata VGG16 vengono pre-elaborate secondo la stessa procedura. Ciò avviene tramite la funzione ***ImageDataGenerator***, utilizzando come funzione di preprocessing *vgg16.preprocess_input*. Gli esempi di addestramento vengono ordinati casualmente (*shuffle=True*) per assicurarsi che ciascun minibatch (di dimensione 32) contenga esempi da tutte le classi. Inoltre, dato che il dataset non ha dimensione molto elevata, si esegue una ***Data Augmentation*** che permette di accrescere i dati di training e validation *"shiftando"* le immagini orizzontalmente e verticalmente per un fattore 0.2 e ribaltandole orizzontalmente.

Il dataset di train viene splittato in Training Set e Validation set utilizzando il parametro *validation_split=0.2* nella funzione *ImageDataGenerator.* Per il training set si definisce un ulteriore funzione di *ImageDataGeneration* (senza eseguire la data augmentation).

3. VGG16

La rete neurale convoluzionale VGG16, sviluppata dal *Visual Geometry Group* di Oxford, ha la seguente architettura, le cui features verranno estratte a diversi livelli in base al modello utilizzato:

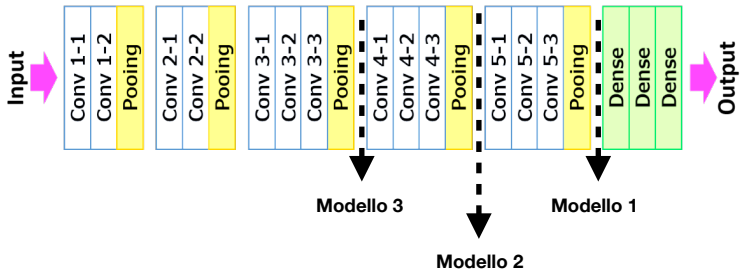

Fig. 1: VGG16 architecture and cut levels

Questa rete viene caricata come *base_net* specificando un *average pooling* e la dimensione di input, pari a 224x224*.* Viene poi costruito un dizionario che mappa i layers ai loro nomi, in modo da rendere più pratico il *"cut"* della rete nei modelli 2 e 3. Ci assicuriamo di "congelare" i parametri della rete in modo che questa funga solamente da feature extractor.

4. Specialized Models

Una volta importata la CNN VGG16 e congelati i suoi layers, si passa alla definizione di un classificatore classico che utilizzerà come input gli output della VGG16.

Il classificatore è lo stesso per tutti e 3 i modelli, in modo da rendere esplicita la comparazione delle performance nell'apprendimento a diversi livelli.

In particolare, agli output della rete VGG16, vengono collegati un layer denso da 512 neuroni con funzione di attivazione *'ReLU'*, un layer di dropout con *rate=0.2* e un layer denso di output di dimensione 10, uguale al numero delle classi, con funzione di attivazione *'softmax'*.

I modelli vengono compilati con funzione di perdita *categorical_crossentropy*, ottimizzatore *RMSprop* (learning rate di default = 0.001) e con metrica *accuracy.*

La fase di training si svolge in 10 epoche con 10 steps per ogni epoca. Quest'ultimo parametro indica quante batch vengono usate usate come input in ogni epoca e serve per limitare il tempo necessario al training.

Il **Modello 1** viene collegato all'output della VGG16 nel layer *global_average_pooling2d*, ultimo layer della rete prima dei layers densi di output. Il modello conta quindi di 23 layers per un totale di 14.982.474 parametri, così suddivisi:

- 20 layers iniziali della rete VGG16 (compreso quello di Input)con 14.714.688 parametri congelati
- 3 layers feed-forward con 267.786 parametri da apprendere

Il modello raggiunge livelli di performance ottimi già alla seconda epoca e raggiunge un accuracy del 94% nel validation set. I risultati nel test set si confermano ottimi, con un accuracy pari al 95%. Di seguito l'andamento delle performances di accuracy e loss nei dataset di training e validation:

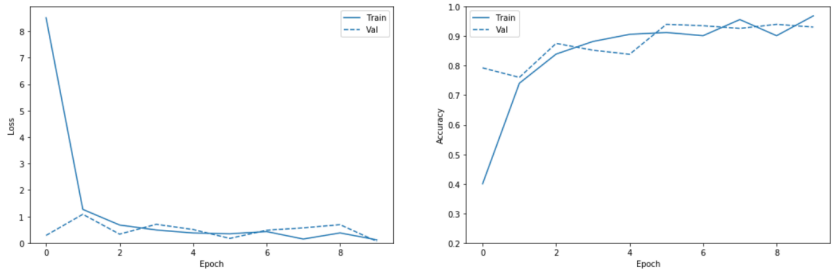

Fig. 2: Model 1 performances on Training and Validation Set

Per collegare il Modello 2 e il Modello 3 alla VGG16 è necessario inserire un layer di pooling intermedio '*Global Average'*, che appiattisca i volumi di output della rete convoluzionale.

Il **Modello 2** utilizza come input l'output del layer di pooling del 4 blocco della VGG16 (*block4_pool*) e ha la seguente architettura:

- 15 layers iniziali della rete VGG16 (compreso quello di Input) con 7.365.264 parametri congelati
- 4 layers feed-forward (il primo di *global average pooling 2d*) con 267.786 parametri da apprendere

Per un totale di 19 layer e 14.982.474 parametri.

La rete non riesce ad apprendere in maniere soddisfacente e l'accuracy in fase di training raggiunge solo il 45% nel validation set. Questo a dimostrazione che le features estratte a tale livello non catturano abbastanza dettagli dell'immagine.

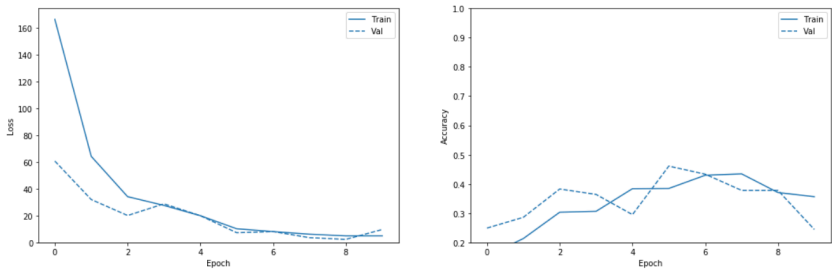

Fig. 3: Model 2 performances on Training and Validation Set

Il **Modello 3** utilizza il layer di pooling del 3 blocco (*block3_pool*) formando la seguente architettura:

- 11 layers iniziali della rete VGG16 (compreso quello di Input) con 1.735.488 parametri congelati
- 4 layers feed-forward (il primo di *global average pooling 2d*) con 136.714 parametri da apprendere

Per un totale di 15 layers e 1.872.202 parametri.

Questo risulta però troppo generico da essere utilizzato come features extractor, come confermato dagli scarsi risultati in fase di learning e testing:

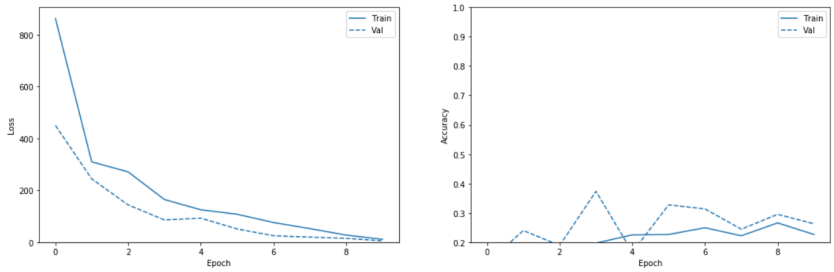

Fig. 4: Model 3 performances on Training and Validation Set

Nella tabella seguente si riassumono i risultati nel test set dei 3 modelli:

	Loss	**Accuracy**
Modello 1	0,0007	0,9595
Modello 2	8,1485	0,3419
Modello 3	6,5590	0,2831