Università degli Studi di Milano Bicocca
Scuola di Scienze
Dipartimento di Informatica, Sistemistica e Comunicazione
Corso di laurea magistrale in Informatica

ASSIGNEMENT 4
Advance Machine Learning

Season Image Recognition

Andrea Guzzo, 761818

Descrizione dell'assignement

L'assignement consiste nell'effettuare finetuning su una rete neurale CNN pre-trainata sul noto dataset di Imagenet.

A tal proposito sono state ottenute immagini appartenenti a quattro classi principali corrispondenti alle stagioni: estate, primavera, autunno, inverno. Utilizzate quindi per effettuare finetuning su una parte della rete neurale.

L'analisi compiuta è reperibile anche al seguente indirizzo su GitHub:
https://github.com/JeyDi/SeasonImageAnalyzer

Dataset

Il Dataset è stato ottenuto a partire dalla costruzione di un crawler appositamente progettato per scaricare le immagini appartenenti alle quattro classi da google image.

Le immagini sono state quindi convertite, sistemate e organizzate in opportune cartelle utilizzate come training set, test set e validation set.

Il codice del crawler e della procedura di data preparation per le immagini sono contenute all'interno della cartella consegnata oppure sempre alla pagina relativa di GitHub.

Il numero di immagini ottenute per ognuna delle 4 classi utilizzate è di 200, che sono state suddivise per il 70% all'interno del trainingset e il 30% per il testset.

Soluzione

La rete neurale utilizzata per il fine tuning è: VGG16 trainata appunto su ImageNet.

La documentazione relativa alla rete utilizzata all'interno della libreria Keras è disponibile al seguente indirizzo: https://keras.io/applications/#vgg16

Le immagini sono state opportunamente trasformate e caricate all'interno dej Jupyter notebook dopo esser state trasformate e suddivise nei rispettivi set di dati.

Inoltre è stata applicata anche una sezione di data augmentation per migliorare la qualità dell'input per favorire la predizione delle immagini rispetto alle classi sul dataset fornito.

A tal proposito è stata utilizzata la funzione ImageDataGenerator messa a disposizione dalle librerie di Keras: https://keras.io/preprocessing/image/

La funzione è stata configurata come segue:

```
train_datagen = ImageDataGenerator(
      rescale=1./255,
      rotation_range=20,
      width_shift_range=0.2,
      height_shift_range=0.2,
      horizontal_flip=True,
      fill_mode='nearest')
```

Inoltre per gli esperimenti è stata utilizzata un image size di 64 x 64 x 3 su immagini appunto RGB.

Per effettuare Fine Tuning sulla rete VGG16 sono stati rimossi dalla rete gli ultimi 4 layers, la configurazione mantenuta nella rete quindi è la seguente:

```
<keras.engine.input_layer.InputLayer object at 0x000002240BA19CC0> False
<keras.layers.convolutional.Conv2D object at 0x000002240BA18668> False
<keras.layers.convolutional.Conv2D object at 0x000002240BA184A8> False
<keras.layers.pooling.MaxPooling2D object at 0x000002240DF15DA0> False
<keras.layers.convolutional.Conv2D object at 0x000002240E146DA0> False
<keras.layers.convolutional.Conv2D object at 0x000002243C3C5320> False
<keras.layers.pooling.MaxPooling2D object at 0x000002243C3DB978> False
<keras.layers.convolutional.Conv2D object at 0x000002243C3DB9E8> False
<keras.layers.convolutional.Conv2D object at 0x000002243C3FCD30> False
<keras.layers.convolutional.Conv2D object at 0x000002243C415080> False
<keras.layers.pooling.MaxPooling2D object at 0x000002243C450940> False
<keras.layers.convolutional.Conv2D object at 0x000002243C4509B0> False
<keras.layers.convolutional.Conv2D object at 0x000002243C46DC18> False
<keras.layers.convolutional.Conv2D object at 0x000002243C48A7B8> False
<keras.layers.pooling.MaxPooling2D object at 0x000002243C4C6908> False
<keras.layers.convolutional.Conv2D object at 0x000002243C4C67F0> True
<keras.layers.convolutional.Conv2D object at 0x000002243C4E9C18> True
<keras.layers.convolutional.Conv2D object at 0x000002243C501780> True
<keras.layers.pooling.MaxPooling2D object at 0x000002243C53B908> True
```

Dove con il flag: True, sono indicati i layer rimossi.

Il modello creato appositamente per rispondere al task è il seguente:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 2, 2, 512)         14714688
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 4100
=================================================================
Total params: 16,816,964
Trainable params: 9,181,700
Non-trainable params: 7,635,264
_________________________________________________________________
```

Sono stati creati quindi una serie di layer fully connected con le seguenti specifiche consultabili anche all'interno del Jupyter notebook fornito:

- Un flatten layer successivo all'ultimo layer rimasto all'interno della rete
- Un layer denso con shape: 1024 utilizzando come funzione di attivazione: Relu
- Un layer di dropout con rate: 0.5
- Il layer di predizione finale sulle 4 classi selezionate con funzione di attivazione: softmax

Il modello è stato quindi trainato utilizzando come funzione di loss: Categorical Crossentropy, mentre come ottimizzatore è stato usato RMSProp in accordo con la documentazione di Keras: https://keras.io/optimizers/

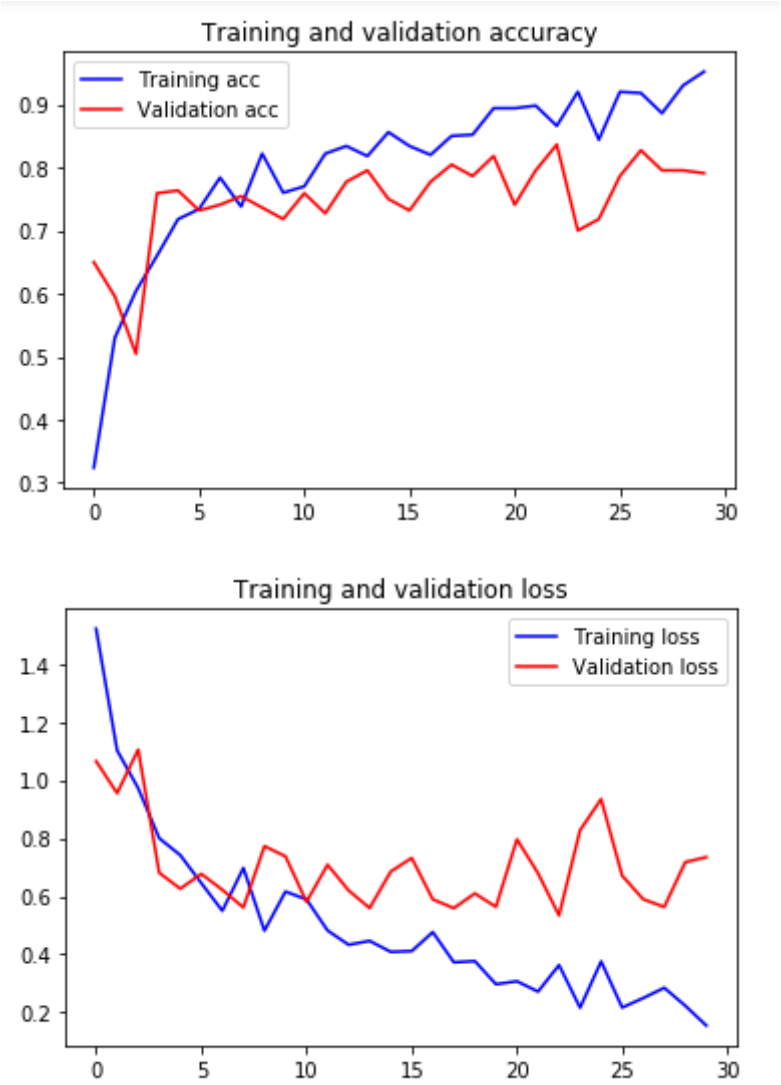
Il numero di epoche utilizzato è: 30, non è stato possibile utilizzare un numero di epoche superiore a causa dell'elevata necessità di performance di calcolo necessarie al training del modello.

Risultati

I risultati migliori ottenuti dopo diverse iterazioni sul dataset costruito con le classi proposte sono i seguenti:

```
Training Accuracy 0.9520000100135804
Training Loss: 0.1553814247250557
Validation Accuracy: 0.7909090898253701
Validation Loss: 0.7346834475809539
```

Inoltre i grafici di training e validation per l'accuratezza e la loss sono i seguenti:



Come possiamo evidenziare dai risultati, le performance del modello sul dataset di validazione non sono molto buone a causa presumibilmente di un elevato overfitting su certe classi di immagini.

È importante considerare che le immagini sono state scaricate in modo automatico dal crawler utilizzando come motore di ricerca: google, il che significa che sicuramente è presenta all'interno dei dati del rumore causato dalla ricerca e dall'utilizzo delle immagini.

Validazione dell'output

Come ultima analisi è stata compiuta una valutazione sull'output delle immagini del modello, su un campione di 220 immagini appartenenti alle diverse classi sono state evidenziate 46 immagini non classificate correttamente dalla rete.

Analizzando quindi empiricamente questi esempi, per alcune immagini è effettivamente complicato distinguere l'appartenza di una classe rispetto ad un'altra.

Per alcune immagini invece l'errore è evidente anche all'occhio umano, ma evidentemente a causa di forme e colorazioni la rete non ha predetto correttamente la classe di appartenenza.

Vengono riportati alcuni esempi significativi:

Original label:autumn\autumn_000031.jpg, Prediction :spring, confidence : 0.925

Original label:spring\spring_000006.jpg, Prediction :summer, confidence : 0.713

Original label:spring\spring_000038.jpg, Prediction :summer, confidence : 0.822

Original label:summer\summer_000120.jpg, Prediction :spring, confidence : 0.928

Original label:winter\winter_000036.jpg, Prediction :summer, confidence : 0.731