Alma Mater Studiorum - Università di Bologna

SCUOLA DI SCIENZE
Corso di Laurea Magistrale in Informatica

TRANSFER LEARNING PROJECT

Specialisation of Deep Learning models

Progetto di Intelligenza Artificiale

Marchesini Matteo
matricola 856336

Sanfelici Matteo
matricola 856403

A.Y. 2018-2019

Introduzione

Il progetto consiste nella generazione di modelli di rete neurali artificiali specializzate su task personalizzati.

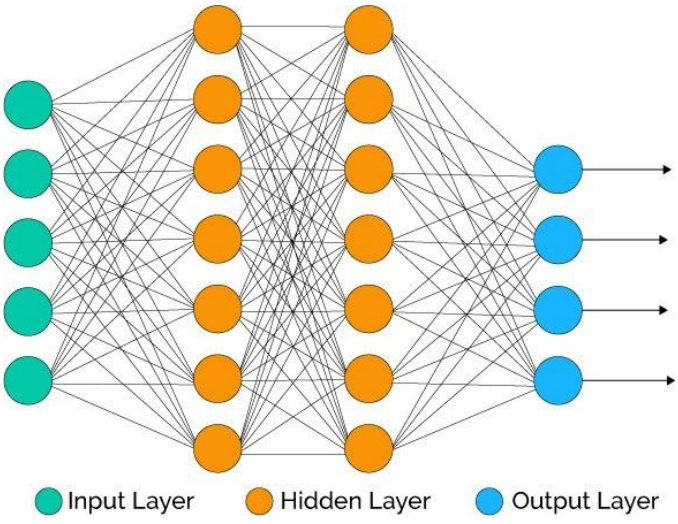
Utilizzando la tecnica di transfer learning, è stato possibile addestrare nuovamente una rete nel riconoscimento e rilevamento di classi non incluse all' interno del modello originale.

L'utente sarà in grado di:

- Scegliere il modello base da utilizzare (object detection o face recognition)
- Inserire nuovi esempi per poi etichettarli con la classe desiderata
- Eseguire il training delle nuove classi inserite
- Salvare il modello generato per l'utilizzo successivo

Transfer Learning

Il Transfer Learning è un metodo di Machine Learning in cui un modello sviluppato per un dato task viene utilizzato come punto di partenza per generare un modello su un secondo problema simile.
È un approccio molto popolare nel deep learning in quanto consente di addestrare Reti Neurali profonde con relativamente pochi dati, che risulterebbero insufficienti per addestrare modelli complessi.



Il Transfer Learning è un'ottimizzazione che consente rapidi progressi o migliori prestazioni durante la modellazione della seconda attività. Generalmente il modello su cui si adotta questa tecnica deve essere quantomeno correlato alla base di conoscenza di quello principale. Per esempio, se ci si basa su di un modello addestrato ad individuare Fiori in un'immagine, si può

specificare tale modello per farlo funzionare per esempio nel riconoscere Frutti.

Ciò che accade nel Transfer Learning è che i pesi di una rete già allenata, vengono trasferiti in una nuova rete e vengono usati per migliorare la sua generalizzazione.

L'idea è quindi di non addestrare da zero una rete neurale, ma basarsi su schemi appresi in attività correlate da dataset molto grandi e trasferire tali schemi su una nuova attività che ha a disposizione un dataset molto limitato.

Transfer Learning è sfruttato principalmente nella Computer Vision e nel Natural Language Processing.

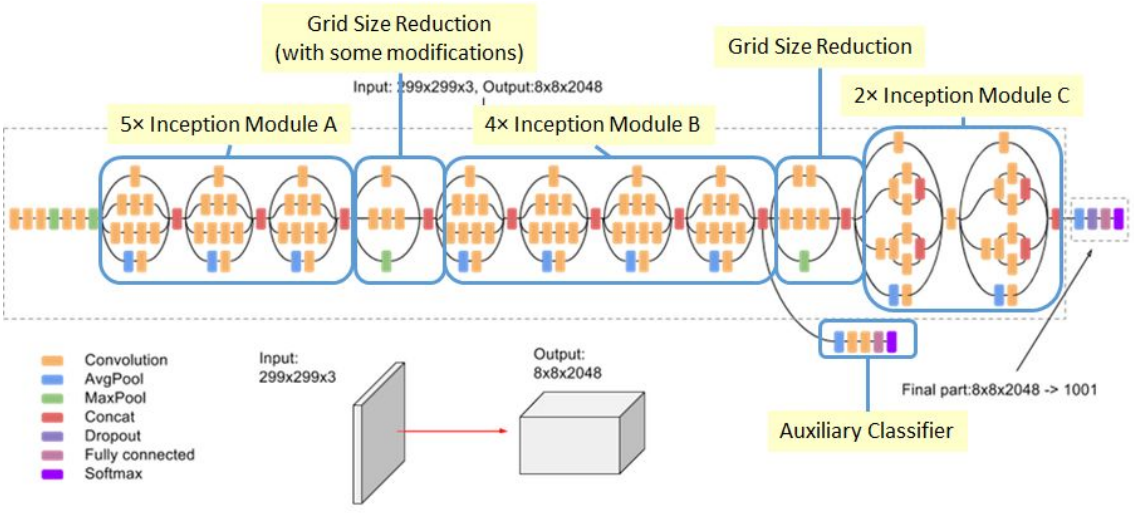
Per sfruttare il modello già pre-allenato, vengono utilizzati i layer iniziali e intermedi della rete principale, mentre si ri-addestrano soltanto gli ultimi strati velocizzando l'allenamento e ottimizzando le operazioni anche per dataset piccoli riuscendo generalmente a migliorare la rete neurale di base.

Il principale vantaggio nell'uso del Transfer Learning si deve al fatto di non dover allenare la rete, risparmiando molto tempo e sopratutto dati, con prestazione anche performanti.

Descrizione del Modello

Modello

Il modello della rete neurale principale su cui è stato applicato il Transfer Learning per fare riconoscimento di Frutti è Inception-v3. Inception è una rete neurale convoluzionale allenato da un dataset costruito da ImageNet.



Una rete neurale convoluzionale è una rete specifica per imparare da dati strutturati a griglia. Questa rete sfrutta un'operazione lineare algebrica chiamata "convoluzione". Per definire una rete convoluzionale, basta utilizzare una convoluzione al posto della moltiplicazione matriciale in almeno un layer.

Convolutional Layers: sono un insieme di filtri (kernel) abbastanza ridotti in dimensione. Ogni filtro viene convoluzionato su tutto l'input, eseguendo il calcolo del prodotto vettoriale tra l'entrate del filtro e l'input e producendo una mappa bidimensionale. Il risultato è una rete che impara solamente dai filtri che vengono attivati quando si identifica una feature nell'input.

Pooling Layers: questi layers hanno il compito di degradare l'immagine. Permettono di ridurre le dimensioni spaziali, ma non la profondità, su una rete neurale di convoluzione, modello, in pratica questo è ciò che si ottiene: avendo meno informazioni spaziali, si ottengono prestazioni di calcolo, e meno informazioni spaziali significano anche meno parametri, quindi meno possibilità di adattamento eccessivo. Per questo layer esistono diverse funzioni non-lineari, Inception utilizza AvgPool e MaxPool.

Concat Layers: servono semplicemente da layer di concatenazione.

Dropout Layer: tale layer ha il compito di evitare l'overfitting di dati, in quanto i neuroni all'interno dei Fully Connected layers sviluppano una co-dipendenza tra loro durante il training.

Fully Connected Layer: rappresenta la parte di ragionamento ad alto della rete. In questo layer i neuroni sono interconnessi ai neuroni del layer precedente. La loro attivazione viene computata con una moltiplicazione matriciale seguita da bias offset.

Softmax Layer: rappresenta l'ultimo layer della rete; permette di eseguire la predizione, assegnando un valore di probabilità a ciascuna classe etichettata del dataset su cui è stato fatto il training della rete. La somma delle probabilità assegnate a ogni classe sarà 1.

Dataset

Il dataset rappresenta una delle componenti principali del progetto, infatti la sua creazione è cruciale per la corretta esecuzione, in quanto è molto importante che le immagini scelte per l'esecuzione siano strettamente attinenti alla categoria di riferimento, in modo da non alterare i risultati, andando a ridurre la precisione del risultato finale.

Il dataset creato è composto da 22 categorie di frutta differenti; per ciascuna di queste categorie sono presenti circa 527 immagini differenti, per un totale di 11.600 immagini.

Le 22 categorie analizzate sono le seguenti:

- Mela Granny Smith
- Mela rossa
- Mela rossa Delicious
- Albicocca
- Banana
- Ciliegia
- Uva blu
- Uva bianca
- Kiwi
- Limone
- Mandarino
- Arancia
- Pera Abate
- Pera Williams
- Pesca
- Prugna
- Melograno
- Lampone
- Fragola
- Pomodoro 1
- Pomodoro 2
- Ananas

Re-training del Modello

Il re-training del modello è avvenuto sulle 22 categorie di frutta descritte sopra. Lo script utile al re-training è chiamato *retrain.py*, il quale permette di caricare il modello già addestrato per poi istruirlo a riconoscere le immagini contenute nel dataset di frutti.

Si rende noto che le immagini di frutti utilizzate per l'addestramento del modello non erano presenti all'interno delle classi ImageNet originali da cui è stata addestrata la rete completa la prima volta.

Bottleneck

La prima fase consiste nell'analizzare le immagini all'interno del dataset, in particolare calcolando e memorizzando i valori di bottleneck per ciascun immagine.

Per "Bottleneck" si intende il layer antecedente al layer finale di output che ha il compito di effettuare la vera e propria classificazione. Il Bottleneck in TensorFlow Hub viene definito "image feature vector", e permette di generare un set di valori sufficienti al classificatore per distinguere, tra le varie classi, quella che gli viene chiesto di riconoscere.

Il risultato quindi deve rappresentare un riassunto significativo e abbastanza compatto delle immagini, così da poter risultare utile alla scelta del classificatore all'interno di un insieme di valori molto piccolo.

Dato che ogni immagine viene riutilizzata più volte durante l'allenamento e il calcolo di ogni bottleneck richiede una notevole quantità di tempo, si è scelto di accelerare il processo memorizzando nella cache i valori dei bottleneck in modo da non essere ripetutamente calcolati. Infatti in totale sono stati memorizzati 11600 bottleneck, uno per ogni immagine.

Vengono memorizzati di default nella directory */tmp/bottleneck*.

Training

Al termine della fase di bottleneck inizia la fase di formazione della rete del livello superiore. I vari step vengono mostrati in output, ciascuno contenenti la precisione di allenamento, la precisione di validazione e l'entropia incrociata.

- Precisione di allenamento: esprime la quantità, in percentuale, di immagini che sono state etichettate con la classe corrente durante il training corrente.
- Accuratezza di validazione: esprime la precisione con la quale un gruppo di immagini viene selezionato da un set diverso ottenuto casualmente.

La differenza sostanziale sta nel fatto che la precisione di allenamento si basa su immagini che la rete è stata in grado di apprendere in precedenza, consentendo una maggior flessibilità nel caso in cui dovesse ricevere dati contenente rumore.

Per misurare le prestazioni effettive della rete è opportuno eseguire delle misurazioni su un dataset non compreso nei dati utili all'addestramento. In tal caso se la precisione di apprendimento risulta elevata ma allo stesso tempo l'accuratezza di validazione è bassa, significa che nella rete sta avvenendo un sovraccarico di

caratteristiche dovuto alla memorizzazione di features durante l'allenamento che non risultano utili al fine dell'apprendimento.
Attraverso l'entropia trasversale, ovvero una funzione di perdita, è possibile osservare il progresso del processo di apprendimento. Infatti l'obiettivo della formazione è di ridurre la perdita il più possibile, così da poter valutare se l' apprendimento della rete funziona correttamente, osservando se la perdita continua a diminuire, ignorando il rumore nel breve termine.

Nel training vengono eseguiti 4000 steps. In ciascuno step vengono scelte dieci immagini casuali dal training set, per poi trovare i bottleneck della cache e portarle al livello finale così da ottenere previsioni. Le previsioni ottenute vengono successivamente confrontate con le etichette effettive per aggiornare i pesi del livello finale attraverso il processo di back-propagation.
Infine viene eseguita una valutazione finale al fine di misurare l'accuratezza dei test effettuati su un dataset esterno a quello di training e di test. In questo modo è possibile stimare nei migliori dei modi le prestazioni di classificazione del modello.

Il modello istruito viene generato nel file "*output_graph.pb*", mentre nel file di testo "*output_labels.txt*" sono presenti le label. Entrambi i file si trovano all'interno della cartella *tmp*.

Test

Una volta completata l'istruzione della rete è necessario verificarne le prestazioni e l'efficienza dei risultati ottenuti. Ciò avviene utilizzando nuove immagini differenti da quelle presenti nel dataset per verificare se il modello è realmente in grado di distinguere la categoria di frutta. Tale verifica avviene attraverso lo script "*label_image.py*", il quale prende in input il grafico, le label create dallo script precedente e la nuova immagine da classificare, restituendo in output la percentuale di accuratezza della categoria di frutta alla quale appartiene l'immagine sottoposta al test.

Steps

1. Il primo step consiste nell'addestrare il modello; ciò avviene lanciando lo script "retrain.py" aggiungendo la directory in cui si trova la cartella del dataset contenente le immagini di frutta utili all'addestramento del modello.

```
>python retrain.py --image_dir {path_directory}\fruits-360\Training
```

2. Ad esecuzione completata dei circa 11600 bottleneck ed effettuati tutti i 4000 steps, è possibile osservare le informazioni relative alle percentuali di apprendimento, alle cross entropy, all'accuratezza di di validazione per i vari step e una percentuale sull'accuratezza finale ottenuta.

```
2019-09-20 11:28:38.548469: Step 3990: Train accuracy = 100.0%
2019-09-20 11:28:38.549471: Step 3990: Cross entropy = 0.040721
2019-09-20 11:28:38.803312: Step 3990: Validation accuracy = 100.0% (N=100)
2019-09-20 11:28:41.005959: Step 3999: Train accuracy = 100.0%
2019-09-20 11:28:41.005959: Step 3999: Cross entropy = 0.033888
2019-09-20 11:28:41.230820: Step 3999: Validation accuracy = 99.0% (N=100)
```

```
I0920 11:29:21.786002  1540 retrain.py:857] Final test accuracy = 99.9% (N=1155)
I0920 11:29:21.885941  1540 retrain.py:1130] Save final result to : /tmp/output_graph.pb
```

3. A questo punto è necessario testare il modello, e per fare ciò viene eseguito lo script *label_image.py,* passando come parametro il grafico generato "output_graph.db", le labels "output_labels.txt", il layer di input "Placeholder", un layer di output "final_result" ed infine il path dell'immagine che si vuole classificare.

```
> python label_image.py --graph=C:\tmp\output_graph.db
  --labels=C:\tmp\output_labels.txt
  --input_layer=Placeholder
  --output_layer=final_result
  --image='.\fruits-360\Test\Pear Williams\1_100.jpg'
```

4. Una volta eseguito il comando precedente, verranno visualizzate i risultati ottenuti in cui è possibile osservare le 5 categorie di frutta che si avvicinano maggiormente all'immagine sottoposta alla classificazione. In questo caso è stata inserita un'immagine di una pera, e come è possibile osservare nell'immagine seguente la percentuale di accuratezza maggiore è data appunto da "pear williams", ovvero un tipo di pera.

```
pear williams 0.9229346
apricot 0.014855751
tomato 1 0.0128554525
pear abate 0.011267323
banana 0.010392737
```

Risultati ottenuti

Attraverso l'utilizzo di Tensorboard sono stati visualizzati e analizzati i risultati ottenuti in fase di apprendimento e di validazione.

Tensorboard dispone di comandi specifici per visualizzare diversi istogrammi relativi al training. Il comando in questione è *tf.summary.histogram* che prende in input i tensori relativi ai comandi da visualizzare durante la computazione.

Gli istogrammi generati evidenziano:

- i valori di Wx+b prima dell'attivazione.

```
tf.name_scope('Wx_plus_b'):
logits = tf.matmul(bottleneck_input, layer_weights) +
layer_biases tf.summary.histogram('pre_activations', logits)
```

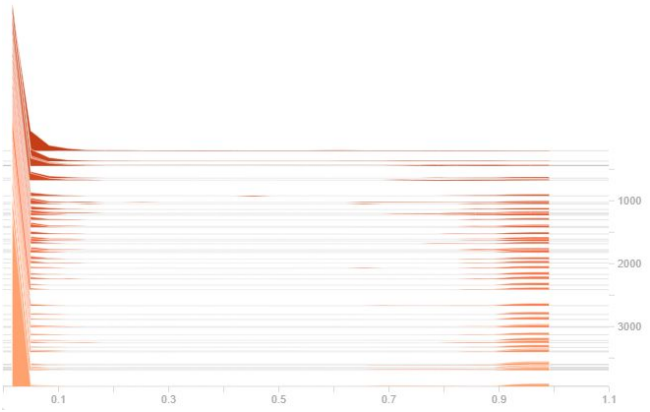
- i valori successivamente all'applicazione della funzione di attivazione

```
final_tensor = tf.nn.softmax(logits, name=final_tensor_name)
tf.summary.histogram('activations', final_tensor)
```

- l'aggiustamento dei pesi durante l'apprendimento per ottimizzare il più possibile l'errore durante i 4000 step di computazione.

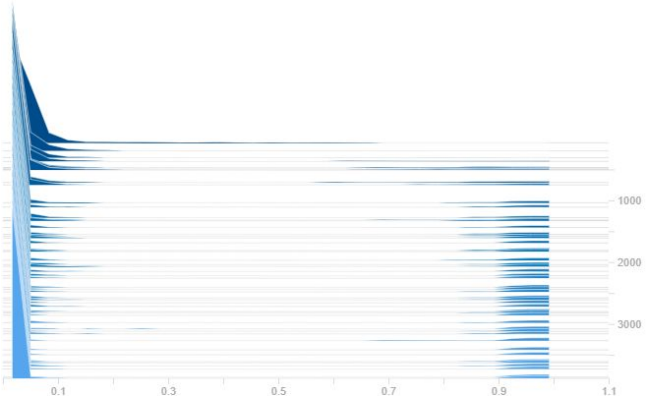
Train

σ(w*X+b) durante il training

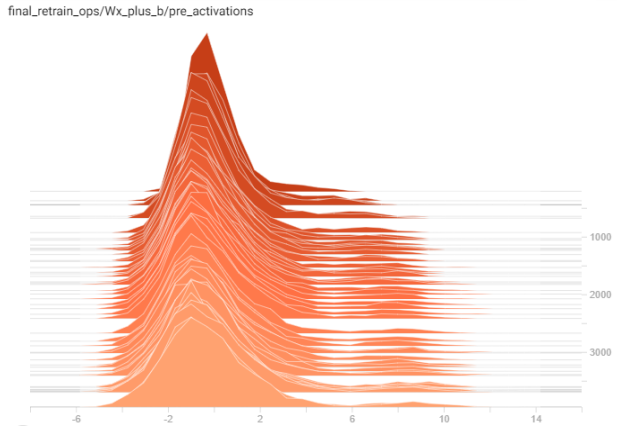


Validation

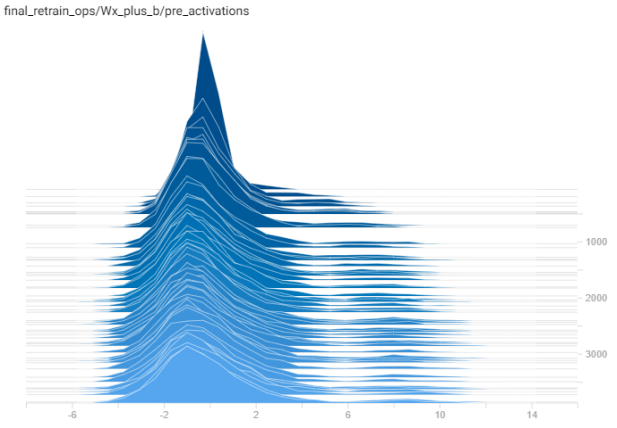
σ(w*X+b) durante la validazione



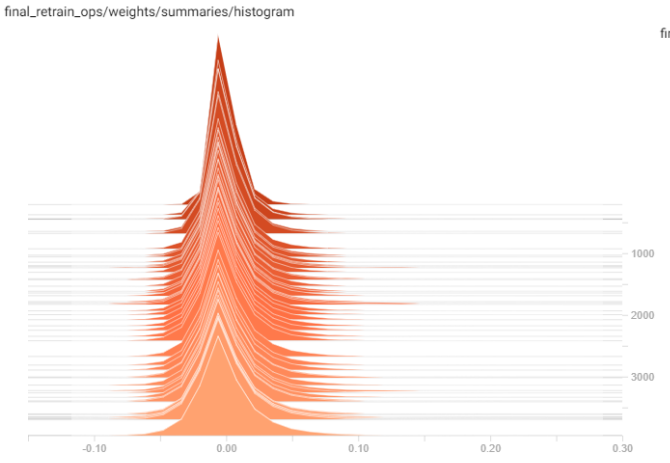
w*X+b pre-attivazione durante il training



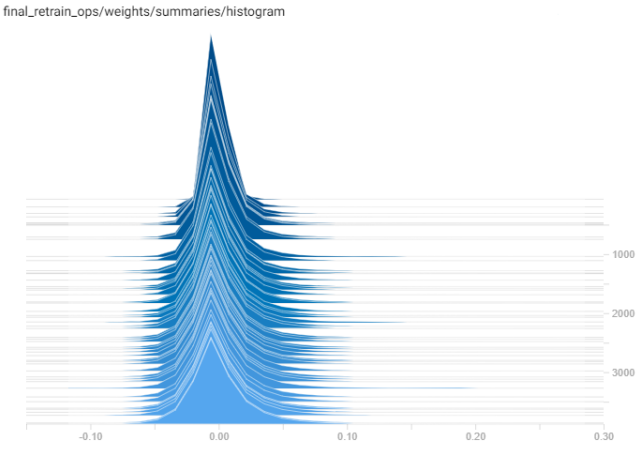
w*X+b pre-attivazione durante la validazione



Ricalcolo dei pesi durante il training



Ricalcolo dei pesi durante la validazione



Conclusioni finali

Terminata la fase di analisi della predizione eseguita mediante Inception v3, dopo aver eseguito il riaddestramento del modello abbiamo ottenuto i seguenti risultati:

- Accuratezza del 99.9%
- Cross entropy pari a 0,03%, quindi prossima allo zero.

Ciò ci ha permesso di sottoporre al modello anche immagini di frutta di categorie simili a quelle utilizzate ottenendo buoni risultati.

Riferimenti

- *Tensorflow*
- *Understanding of Convolutional Neural Network (CNN) — Deep Learning*
- Interpreting Deep Learning Models for Computer Vision