# Assignment 3 (Advanced Machine Learning)

Davide Sangalli - Matricola: 848013

November 11, 2019

**Abstract**

In questo assignment si tratta il problema della classificazione delle immagini, in particolare del riconoscimento di cifre scritte a mano, usando l'*MNIST dataset*

## 1 Preprocessing

Dopo aver importato il dataset, esso è stato diviso in 3 parti: training set, validation set e test set. Il validation set è stato ricavato splittando ulteriormente il training set, usandone il 20% per fare validation.
Le immagini sono state normalizzate (dividendole per 255) ed è anche stato effettuato un *reshape*, in modo tale che ogni immagine fosse definita da tre valori: altezza, larghezza e canale. Quest'ultimo è pari ad 1, in quanto le immagini sono in scala di grigi. Altezza e larghezza sono entrambe pari a 28. Perciò un'immagine ha questa forma: *(28,28,1)*.
Il reshape è stato inoltre necessario, poichè la forma *(height,width,channel)* è quella predefinita per poter elaborare le immagini con Keras.
Infine, le *labels* sono state trasformate tramite la codifica *one-hot encoding*, per poter essere utilizzate correttamente dalla rete neurale.

## 2 Convolutional Neural Network

### 2.1 Architettura della rete

La rete neurale utilizzata per questo problema è di tipo convoluzionale (*CNN*).
In particolare è così strutturata:

- un *input layer* con *shape=(28,28,1)*, come descritto precedentemente
- un *convolutional hidden layer* con un banco di 16 filtri 3x3 e funzione di attivazione *relu*
- un *MaxPooling layer* di 2x2

- i punti 2 e 3 si ripetono, andando così a formare un altro layer convoluzionale ed uno di MaxPooling
- un *flatten layer*, necessario per effettuare la conversione delle immagini da un formato bidimensionale ad uno monodimensionale
- un *output layer*, con 10 neuroni, ognuno corrispondente ad una classe e con funzione di attivazione *softmax*
- il modello è stato compilato utilizzando *adam* come ottimizzatore (con *learning_rate*=0.001), *categorical_crossentropy* come funzione di perdita e come metrica l'*accuracy*
- è stato utilizzata una *batch_size* di 128 e un numero di epoche pari a 15

NB: In fase di costruzione del modello erano stati aggiunti dei *dropout layers* con un *dropout_rate* di 0.1, ma sono stati successivamente eliminati perchè si è notato che inficiavano sull'accuracy del modello, che, senza di essi, performa meglio.

## 2.2 Calcolo del numero dei parametri della rete

Di seguito viene riportato il summary della rete neurale:

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 16)        160
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 13, 13, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 16)        2320
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 5, 5, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                4010
=================================================================
Total params: 6,490
Trainable params: 6,490
Non-trainable params: 0
```

Il modello ha un totale di 6490 parametri, che rispettano il vincolo imposto di avere meno di 7500 parametri.
Eseguendo i calcoli si ha che:

- per il primo layer convoluzionale il numero totale di parametri è: (((3*3)+1)*16)=160, dove (3*3) è la dimensione del filtro, 1 è il bias e 16 è il numero totale di filtri di questo layer
- con un ragionamento analogo al primo, si ricavano i parametri del secondo layer convoluzionale; essi sono pari a: (((3x3)*16+1)*16)=2320 poichè il primo layer convoluzionale ha 16 filtri
- i layer di MaxPooling non hanno parametri
- il layer Flatten non ha parametri
- il layer fully-connected (Dense) ha un numero di parametri pari a: (400+1)*10=4010, dove 400 è la dimensione del flatten layer precedente, 1 è il bias e 10 è il numero di classi del problema

Perciò la rete ha un numero totale di parametri pari a: 160+2320+4010=6490 ($< 7500$).

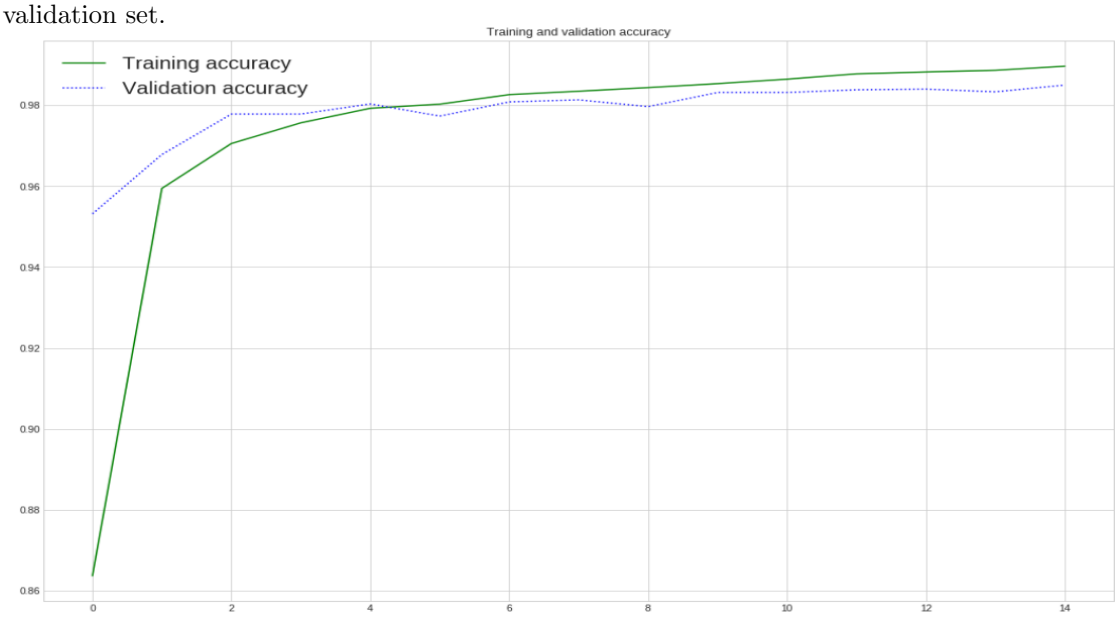
### 2.3 Note sull'output shape dei layers della rete

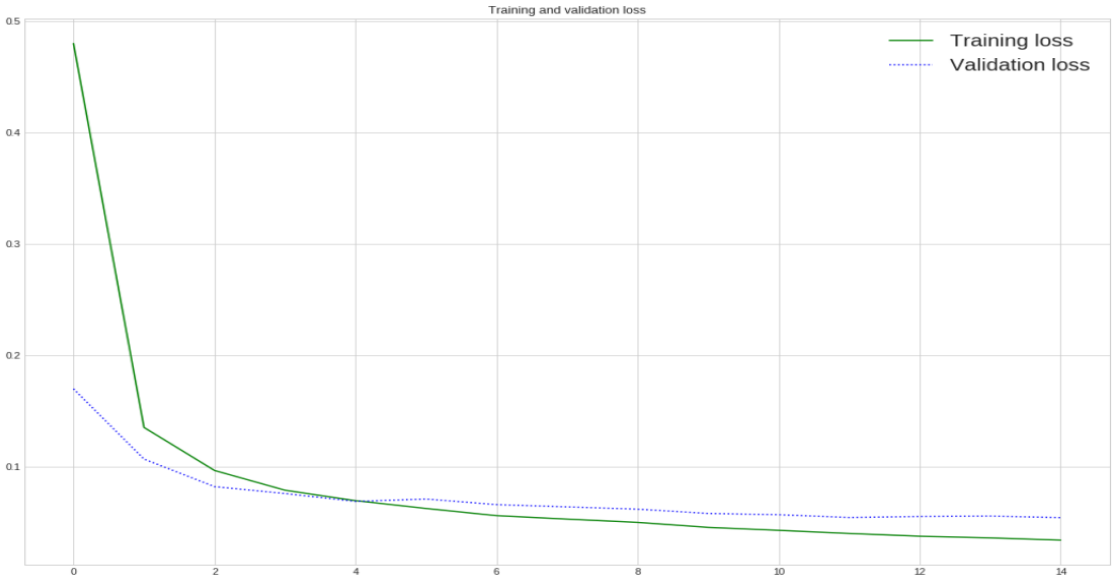
L'immagine precedente mostra anche gli output shape di ogni layer della rete. Di seguito verrà fornita una spiegazione:

- l'output shape del primo layer è di (26,26,16). Poichè è stato applicato un *no_padding* (di default nei layers Conv2D di Keras) e un filtro 3x3 al primo layer convoluzionale, l'immagine risultante in output è una 26x26, con 16 filtri applicati.
- poichè è stato applicato un MaxPooling di 2x2 sull'immagine risultante dal primo layer convoluzionale, l'output shape dell'immagine è ora di 13x13 (la dimensione viene dimezzata). 16 sono ancora una volta i filtri che sono stati applicati nel layer precedente.
- con ragionamento analogo a quello fatto per il primo layer convoluzionale, l'output shape del secondo layer convoluzionale è un'immagine 11x11, sempre con 16 filtri applicati.
- viene nuovamente applicato un MaxPooling di 2x2 e la dimensione dell'immagine è dunque una 5x5 ora (sempre con 16 filtri applicati nel layer precedente).
- il Flatten layer ha una dimensione di output pari a 5*5*16=400, ovvero comprime l'output shape del layer convoluzionale precedente in un vettore monodimensionale.
- infine, l'output shape dell'ultimo layer (Dense) è di 10, corrispondente alle 10 classi del problema.

# 3 Risultati

Di seguito vengono riportati i grafici di accuracy e loss del training set e del validation set.




Da questi grafici si può notare che non c'è overfitting del modello, in quanto il training set raggiunge un'accuratezza del 99% e il validation set del 98%. Analogamente, la loss del training set e del validation set hanno circa lo stesso andamento.

Questi plots confermano quindi che il modello scelto è buono.

Vengono riportati, infine, per completezza, nell'ordine, i *classification_reports* di training set, validation set e test set:

TRAINING SET CLASSFICATION REPORT

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 0.99 | 5299 |
| 1 | 1.00 | 0.99 | 0.99 | 6088 |
| 2 | 0.99 | 0.98 | 0.99 | 5386 |
| 3 | 0.99 | 0.99 | 0.99 | 5542 |
| 4 | 1.00 | 0.99 | 0.99 | 5262 |
| 5 | 1.00 | 0.99 | 0.99 | 4870 |
| 6 | 0.99 | 1.00 | 0.99 | 5338 |
| 7 | 0.99 | 0.99 | 0.99 | 5632 |
| 8 | 0.98 | 0.99 | 0.98 | 5266 |
| 9 | 0.97 | 0.99 | 0.98 | 5317 |
| accuracy | | | 0.99 | 54000 |
| macro avg | 0.99 | 0.99 | 0.99 | 54000 |
| weighted avg | 0.99 | 0.99 | 0.99 | 54000 |

VALIDATION SET CLASSIFICATION REPORT

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.98 | 0.99 | 624 |
| 1 | 1.00 | 0.99 | 1.00 | 654 |
| 2 | 0.98 | 0.97 | 0.98 | 572 |
| 3 | 0.99 | 0.99 | 0.99 | 589 |
| 4 | 0.99 | 0.98 | 0.99 | 580 |
| 5 | 0.99 | 0.98 | 0.99 | 551 |
| 6 | 0.99 | 0.99 | 0.99 | 580 |
| 7 | 0.98 | 0.99 | 0.99 | 633 |
| 8 | 0.96 | 0.99 | 0.97 | 585 |
| 9 | 0.97 | 0.99 | 0.98 | 632 |
| accuracy | | | 0.98 | 6000 |
| macro avg | 0.99 | 0.98 | 0.98 | 6000 |
| weighted avg | 0.99 | 0.98 | 0.99 | 6000 |

TEST SET CLASSIFICATION REPORT

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 1.00 | 0.99 | 0.99 | 980 |
| 1 | 0.99 | 0.99 | 0.99 | 1135 |
| 2 | 0.99 | 0.98 | 0.98 | 1032 |
| 3 | 0.98 | 0.99 | 0.99 | 1010 |
| 4 | 0.99 | 0.98 | 0.99 | 982 |
| 5 | 0.99 | 0.98 | 0.99 | 892 |
| 6 | 0.99 | 0.99 | 0.99 | 958 |
| 7 | 0.97 | 0.99 | 0.98 | 1028 |
| 8 | 0.97 | 0.99 | 0.98 | 974 |
| 9 | 0.98 | 0.99 | 0.98 | 1009 |
| accuracy | | | 0.99 | 10000 |
| macro avg | 0.99 | 0.99 | 0.99 | 10000 |
| weighted avg | 0.99 | 0.99 | 0.99 | 10000 |

Si può notare che l'accuracy sul training set è del 99%, sul validation set del 98% e sul test set ancora del 99%, ad ulteriore conferma che il modello costruito generalizza bene su dati mai visti prima ed è quindi buono.