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

ASSIGNEMENT 2
Advance Machine Learning

Letters recognition

Andrea Guzzo, 761818

Descrizione dell'assignement

L'assignement consiste nella predizione di lettere rappresentate in scala di grigio dalla A alla J.

Il Dataset contiene sia immagini di train che di test, sia un secondo dataset su cui è stata realizzata una predizione dopo aver costruito il modello.

Soluzioni proposte

Per completare il task assegnato, vengono proposte due differenti soluzioni che hanno portato a risultati differenti:

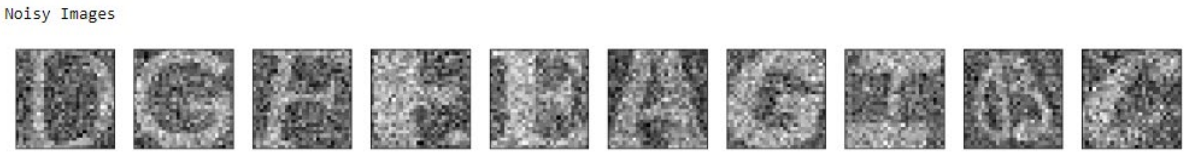
1. Rete neurale deep feed forward (all'interno del notebook: experiment-normal.ipynb)
2. Autoencoder (all'interno del notebook: experiment-autoencoders.ipynb)

Entrambe le soluzioni sono state ottenute utilizzando Keras su Tensorflow backend su CPU

La creazione dei modelli è stata completata per entrambi, i pesi e i relativi modelli sono stati esportati anche in formato json.

I modelli sono all'interno della cartella di consegna e sono liberamente utilizzabili, inoltre per la rete neurale deep feed forward è stato costruito anche un file chiamato: result-normal.txt contentente le predizioni effettuate dal classificatore multiclasse.

Per entrambe le soluzioni è stato inoltre calcolato e utilizzato per alcune sperimentazioni lo stesso dataset con aggiunto del rumore sulle immagini con un noise factor pari a 0.5.



L'analisi compiuta è reperibile anche al seguente indirizzo su GitHub:

https://github.com/JeyDi/Digits-NeuralNet

Soluzione 1: Feed Forward network

Per la prima soluzione è stata costruita una Deep Neural Network Feed Forward utilizzando come funzione di loss: Categorical Cross Entropy

Come ottimizzatore è stato utilizzato: Adam.

La rete è quindi composta da due layers densi, uno composto da 32 unità che utilizza come funzione di attivazione: Relu (Recrified Linear Unit), mentre il layer finale composto da 10 unità che si occupano di prevedere le lettere all'interno del dataset utilizzando come funzione di attivazione: Sigmoid.

In questo modo effettuando un reshape delle immagini in formato matriciale e normalizzando tutti i valori tra 0 e 1 appiattendo inoltre le immagini in formato 28x28 in vettori di dimensione 784 è stato possibile ottenere una classificazione sufficiente per ottenere un buon grado di performance del nostro modello.

La rete inoltre utilizza un batch_size di 256 records, un validation split di 0.2 e un numero di epoche pari a 50 che vengono regolarizzate da una funzione di early stopping automatica con un delta pari a 0.0001 che mediamente arresta il numero di epoche durante il training a 18

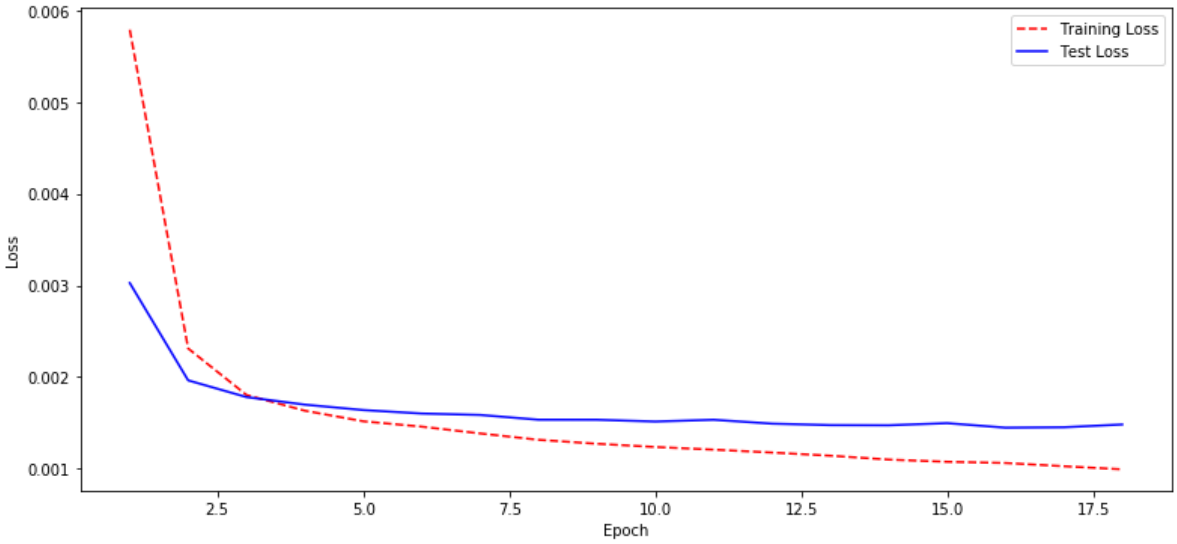
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_57 (Dense)             (None, 32)                25120
_________________________________________________________________
dense_58 (Dense)             (None, 10)                330
=================================================================
Total params: 25,450
Trainable params: 25,450
Non-trainable params: 0
_________________________________________________________________
```

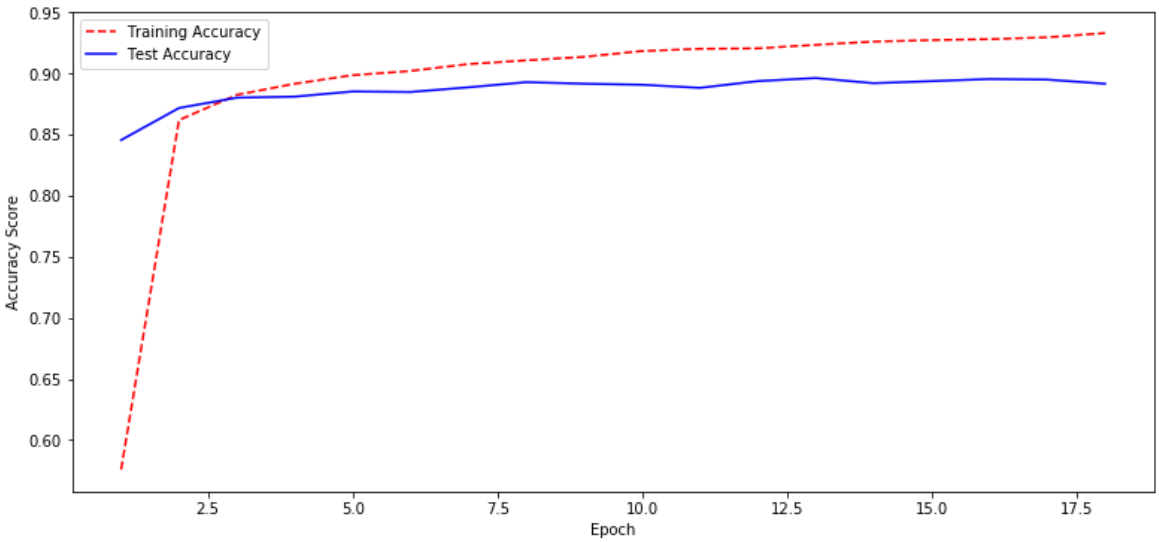
Il risultato effettuato sul dataset di training è si attesta a 93% dei dati, con una loss inferiore all'1%

```
Epoch 18/50
9463/9463 [==============================] - 0s 18us/step - loss: 9.8905e-04 - acc: 0.9328 - val_loss: 0.0015 - val_acc: 0.8914
Epoch 00018: early stopping
```

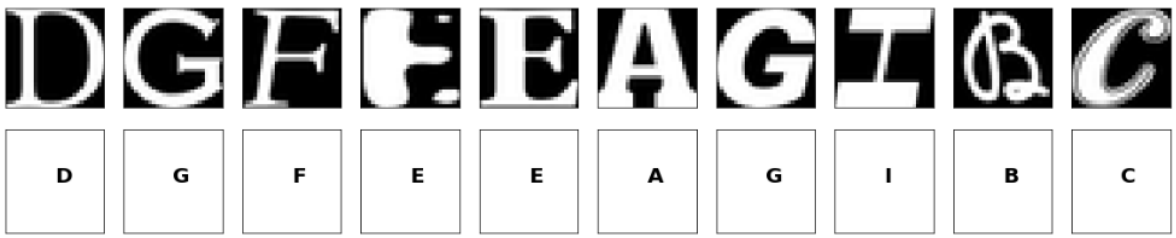
I risultati del test set sono inferiori, probabilmente per la poca disponibilità di dati.

I grafici ottenuti di accuratezza e di loss tra training e test hanno fornito i seguenti risultati:





Un esempio dei risultati ottenuti invece dalla predizione delle nuove labels sul test set fornito come secondo input dell'assignement sono i seguenti:



Con la soluzione proposta è possibile quindi effettuare una buona predizione delle lettere con un buon grado di confidenza.

Soluzione 2: Autoencoder

Per la soluzione proposta utilizzando un autoencoder è stato utilizzata una grandezza della rappresentazione a 32 floats con un fattore di compressione di 24.5.

La rete è stata costruita utilizzando due layers di encoding sempre con una ReLU e con 32 units.

Per il decoding invece è stata utilizzata come funzione di attivazione una sigmoide con 784 units pari al numero di floats delle immagini in input.

La rete inoltre utilizza come ottimizzazione adadelta e come loss function: binary crossentropy in quanto le immagini sono pixels normalizzati da 0 a 1.

Il batch size è sempre impostato a 256 come nell'esempio precedente, abbiamo inoltre utilizzato nuovamente la stessa funzione di early stop.

I risultati rispetto alla rete presentata precedentemente sono inferiori, presumibilmente proprio per le caratteristiche strutturali degli autoencoders.

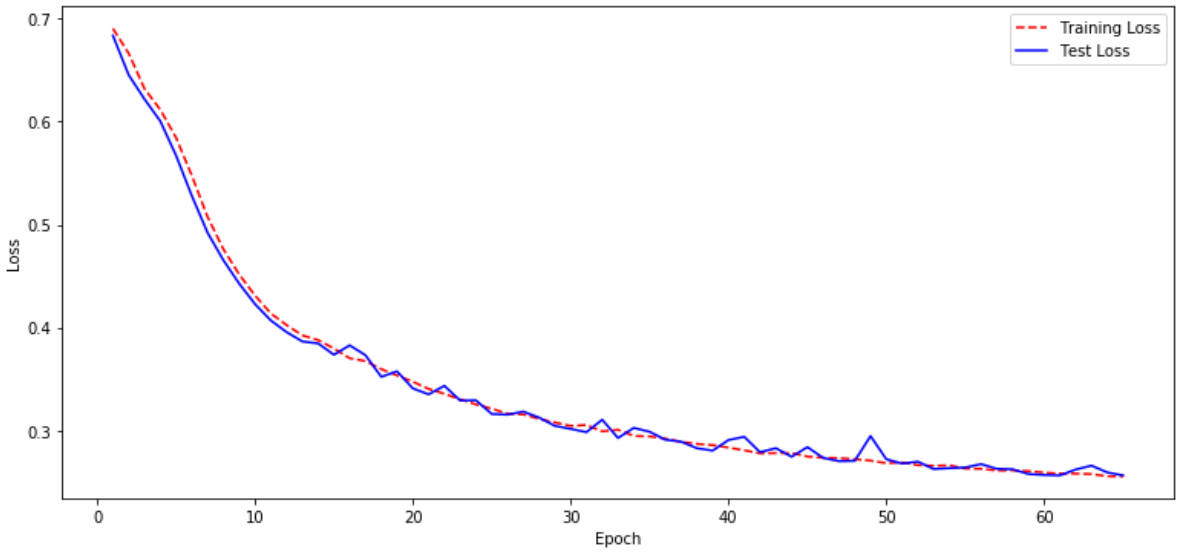
Le performance misurate dopo diversi training raggiungono a fatica un'accuratezza del 60% con una loss function che non scende mai sotto al 21%.

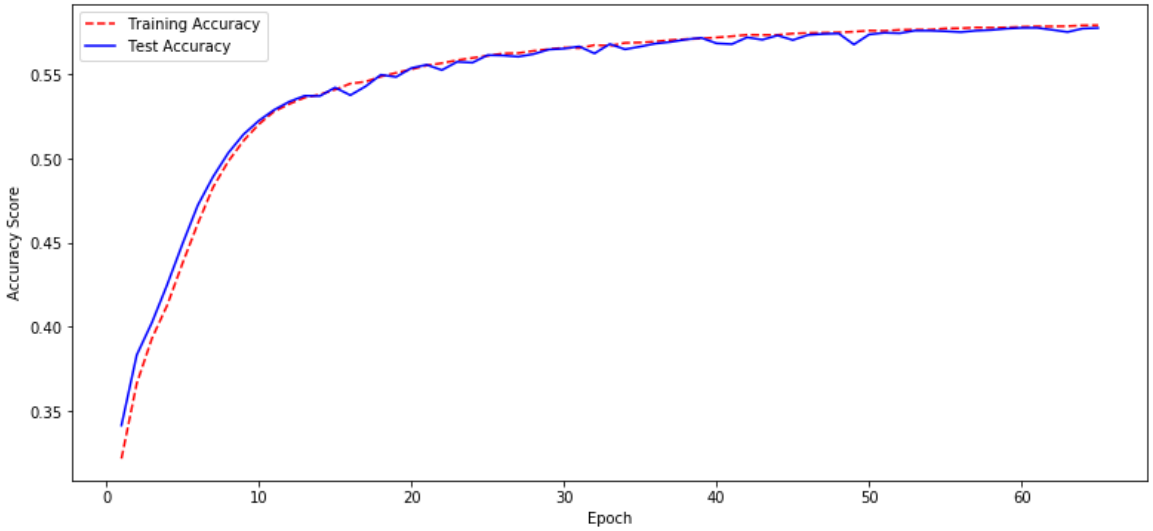
Per quanto riguarda le epoche, grazie all'early stopping arriviamo a in media a circa 65 epoche per il training.

Le caratteristiche della rete sono quindi riassunte nel seguente schema:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_7 (InputLayer)         (None, 784)               0
_________________________________________________________________
dense_10 (Dense)             (None, 32)                25120
_________________________________________________________________
dense_11 (Dense)             (None, 32)                1056
_________________________________________________________________
dense_12 (Dense)             (None, 784)               25872
=================================================================
Total params: 52,048
Trainable params: 52,048
Non-trainable params: 0
```

Visualizzando i grafici di loss e accuratezza sul training e sul test set possiamo notare qualche overfitting al superamento delle 10 epoche sul test set.





Oltre al training della rete è stata implementata anche un encoding e un decoding sui dati di test per ricostruire le sequenze in base al modello.

Il risultato è il seguente, dove nella prima riga sono presenti alcune immagini dei dati di input, mentre nella seconda riga le lettere ricostruite dalla rete trainata.