Assignment 2: Advance Machine Learning

Davide Brinati

Matricola: 771458

1 Introduzione

Il dataset oggetto di questo assignment, riguarda diverse immagini in bianco e nero di lettere, di dimensione 28x28 pixel. Tale dataset risulta essere composto da un train set, formato da 11829 immagini con un label, e da un test set, comprendente 5070 immagini senza label. Di seguito è possibile osservare 10 immagini appartenenti al training.

In questo assignment si utilizzeranno diversi modelli di Autoencoding, per cui sulle immagini verranno applicati metodi di Deep Learning non supervisionati; successivamente tali modelli verranno estratti ed utilizzati in maniera supervisionata, ovvero si addestrerà un classificatore sul training per poi utilizzarlo per prevedere i label mancanti del test.

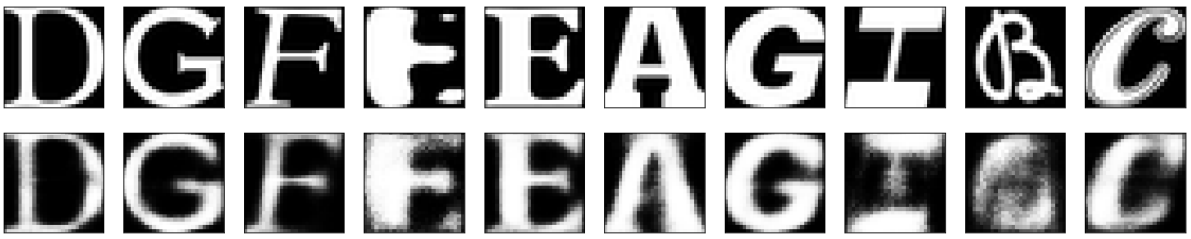
2 Autoencoding

In questa sezione verrà descritta la metodologia con cui gli autoencoder sono stati applicati. Inizialmente è stato applicato un semplice autoencoder, successivamente si è optato per un autoencoder più elaborato, con più strati intermedi, ed infine è stato utilizzato un autoencoder convoluzionale.

Simply Autoencoder

Il primo autoencoder utilizzato risulta essere molto semplice, in quanto formato da un solo strato di codifica, con funzione di attivazione relu, e uno di decodifica, con funzione di attivazione sigmoid. La dimensione di codifica è stata impostata pari a 32, in quanto viene applicato un fattore di compressione pari a 24.5 sull'input, il quale risulta essere un'array di 784 valori (matrice (28, 28) trasformata in array (784,1)). Lo strato di decodifica restituisce un array di 784 valori, e sarà l'output dell'autoencoder. Tale modello è stato allenato su 50 epoche.

Di seguito è possibile osservare come agisce l'autoencoder sulle immagini. In alto sono visualizzate 10 immagini del test. In basso è possibile osservare l'output del modello relativo alla soprastante immagine.



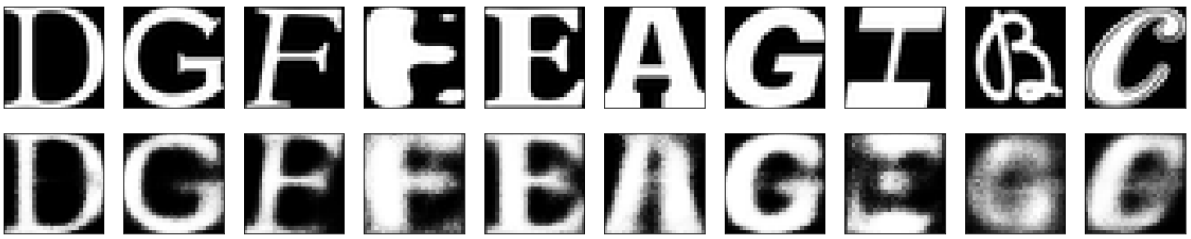
Deep Autoencoder

Dopo aver utilizzato un modello di Autoencoding semplice, formato da due soli strati, si implementa ora un modello più complesso, formato da più layer. Tale modello riceve in input array composti 784 valori, come il modello precedente, ma risulta essere formato da tre strati di codifica e tre di decodifica.

```
input_img = Input(shape=(784,))
encoded = Dense(128, activation='relu')(input_img)
encoded = Dense(64, activation='relu')(encoded)
encoded = Dense(32, activation='relu')(encoded)

decoded = Dense(64, activation='relu')(encoded)
decoded = Dense(128, activation='relu')(decoded)
decoded = Dense(784, activation='sigmoid')(decoded)
```

Sopra troviamo riportato il codice: nei tre layer di codifica l'input viene ridotto, fino ad avere una dimensione di codifica pari a 32. Nei tre layer di decodifica la dimensione viene riportata a quella di partenza. Per tutti gli strati la funzione di attivazione usata è la relu, a parte l'ultimo strato di decodifica al quale è stata applicata la sigmoid. Il modello è stato allenato per 50 epoche. Di seguito è possibile osservare e confrontare l'output di tale autoencoder con le immagini originali del test. (Vengono riportate solo 10 immagini)

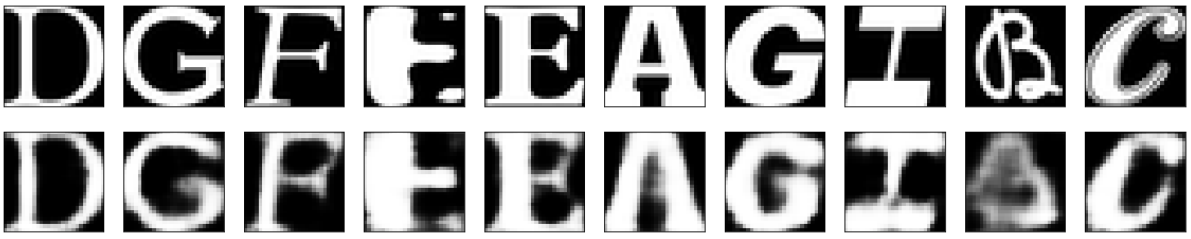


Si possono notare ovvie differenze anche fra gli output di questo modello e di quello prcedente (modello semplice).

Convolutional Autoencoder

L'ultimo modello utilizzato è un Convolutional Autoencoder. Dato che abbiamo a che fare con immagini, risulta ragionevole utilizzare reti convoluzionali come codificatore e decodificatore.

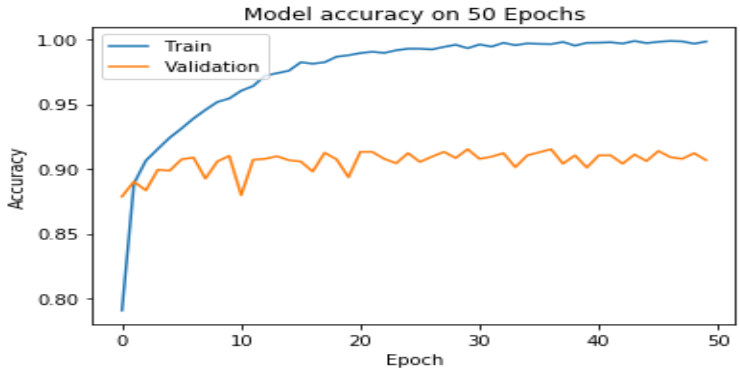
Il codificatore è composto da strati di convoluzione e da strati di maxpooling, necessari per la riduzione spaziale. Il decodificatore è formato da strati convoluzionali e da strati UpSampling2D. Tutti gli strati hanno come funzione di attivazione la relu, a parte per l'ultimo strato del decodificatore, che ha la sigmoid. Di seguito è possibile osservare il confronto su 10 items, fra le immagini del test e il relativo output dell'autoencoder.

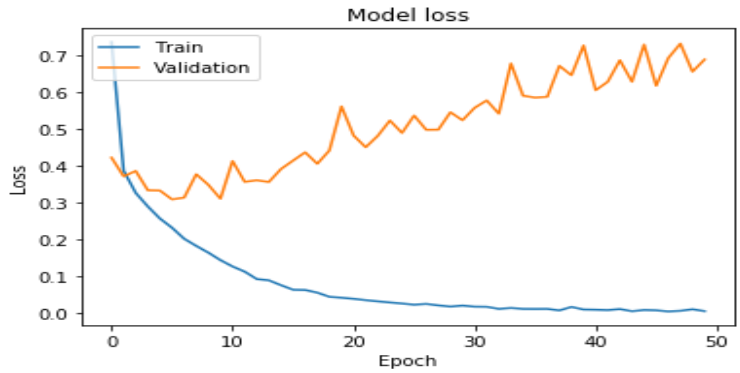


3 Classificazione

In questa sezione verrà trattata la parte di apprendimento supervisionato: si cercherà di elaborare un classificatore che, apprendendo dal training, riesca a predirre le classi del test col minore tasso d'errore possibile. A tal fine il training è stato diviso in train e validation, con uno split pari allo 0.25. Per classificare si utilizzeranno i modelli sviluppati precedentemente per le task di autoencoding: all'encoder verranno aggiunti ulteriori strati di output, necessari per creare un modello di classificazione multi-class. Si ricorda che le immagini rappresentano delle lettere scritte a mano, in una scala di grigi, dalla A alla J. Sono presenti quindi 10 classi. I modelli utilizzati sono quelli derivanti dal *Deep Autoencoder* e dal *Convolutional Autoencoder*. Il modello semplice non è stato utilizzato per la classificazione.

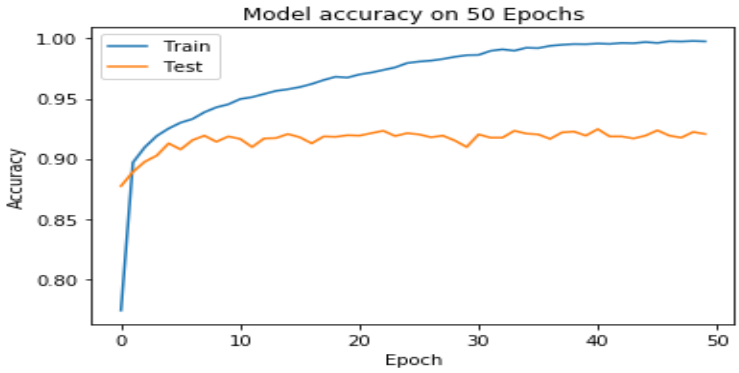
Il modello derivante dal **Deep Autoencoder** è stato addestrato per 50 epoche sul train, e testato sul validation attrettante vole. Sotto è possibile osservare l'andamento dell'accuracy e della loss su 50 epoche. Alla rete sono stati aggiunti due strati, l'ultimo dei quali ha 10 neuroni e come funzione di attivazione *softmax*. Il modello utilizza come funzione di perdita la *categorical crossentropy* e l'ottimizzatore è *adadelta*. Inoltre si è utilizzato il "callback" *best model*, il quale salva i pesi del modello che raggiunge un accuracy più alta sul validation e li salva in un file.

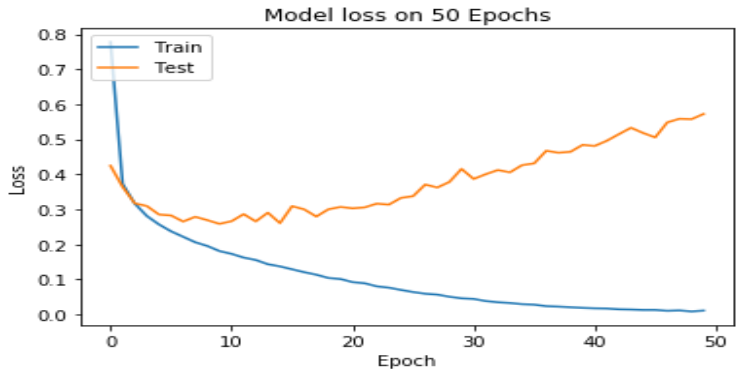




Il modello derivante dal **Convolutional Autoencoder** è stato addestrato per 50 epoche. All'encoder utilizzato per la task di autoencoding convoluzionale è stato aggiunto uno strato *flatten*, con il compito di appiattire le matrici 28x28 in array di dimensione 784; infine sono stati aggiunti due strati, l'ultimo dei quali ha 10 neuroni, come il numero delle possibili classi, e funzione di attivazione *softmax*. La funzione di perdita utilizzata è ovviamente la *categorical crossentropy* e l'ottimizzatore è *adadelta*. Anche in questo caso si è utilizzato il callback *best model*.

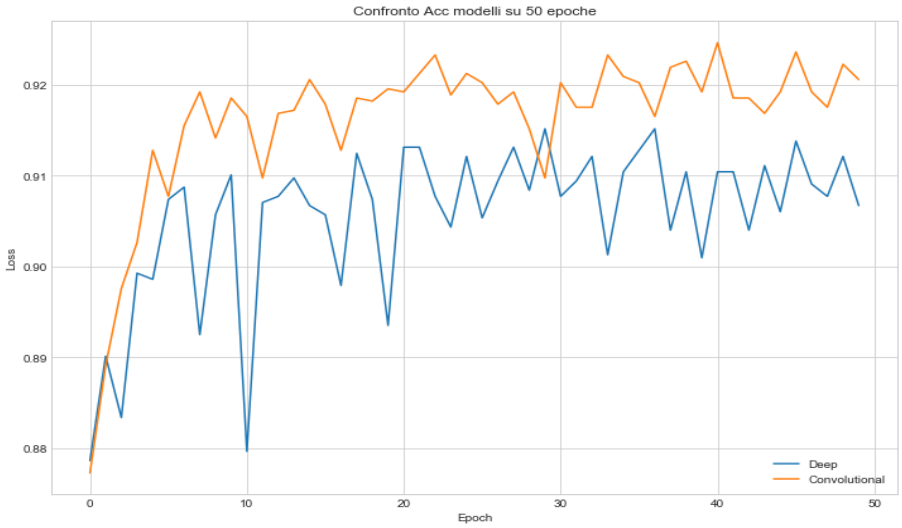
Di seguito è possibile osservare l'andamento di accuracy e loss su 50 epoche.





4 Conclusioni

In questa sezione si effetturà un confronto fra modelli, in modo da **scegliere il modello con performance migliori sul validation**, e utilizzarlo per prevedere sul test. Di seguito è possibile osservare l'andamendo dell'accuracy sul validation dei due modelli, su 50 epoche di apprendimento.



Risulta evidente che il modello convoluzionale ha performance migliori rispetto all'altro, pertanto si utilizzerà la rete convoluzionale per generalizzare sul test.