Report dell'Assignment 3 di Advanced Machine Learning

Paolo Mariani 800307

Novembre 2019

1 Introduzione

Lo scopo dell'assignment è quello implementare una rete neurale di tipo CNN (*Convolutional Neural Network*) per svolgere il compito di classificazione di immagini rappresentanti numeri scritti a mano.
Il dataset da cui si prelevano le osservazioni è il ***MNIST Database of Handwritten Digits***, risulta composto da 70000 osservazioni; ciascuna osservazione viene caricata nell'ambiente di programmazione con la dimensione 28x28 pixel: è interpretata da esso come una matrice di 28 righe e 28 colonne tale per cui ogni elemento descritto dall'incrocio riga-colonna (pixel) possiede un valore espresso in scala di grigi e varia da 0 a 255. Si mostrano alcune immagini di esempio per comprendere il contenuto del Dataset:

Figura 1: Esempio di osservazioni appartenenti al Dataset

1.1 Preprocessing

Sul dataset sono state effettuate alcune operazioni di Preprocessing quali:

- Normalizzazione del colore da [0, 255] a [0, 1].
- Split dei dati in Training Set (60000 osservazioni) e Test Set (10000 osservazioni) effettuato in maniera automatica dalla funzione *mnist.load_data()*
- Applicazione della *codifica One-Hot* alla variabile target del problema: da cifra intera (0-9) viene rappresentata come vettore binario di lunghezza 9 composto da elementi pari a 0 ad esclusione di quello situato in posizione corrispondente al valore dell'attributo, il quale assume valore 1.

2 Architettura della Rete Neurale

La rete neurale che deve svolgere il compito di classificazione è di tipo CNN, ossia una rete ispirata al funzionamento della corteccia visiva che permette di riconoscere oggetti all'interno del campo visivo grazie ad una gerarchia di *feature detectors*.
Il processamento di una rete CNN è identico alle classiche reti Feed-Forward, in quanto sono reti neurali a cui si aggiunge l'importante operazione di convoluzione; la tipologia di strati implementati in esse permette di evvettuare operazioni di ***convoluzione*** e ***sintesi*** delle caratteristiche principali che compaiono all'interno dell'immagine, tramite l'applicazione di filtri convoluzionali (chiamati anche filter, mask, kernel, ...).
Le reti neurali convoluzionali classiche implementano una architettura composta da diversi tipi di operazioni:

- di *Convoluzione* che si occupano di riconoscere i pattern contenuti nell'immagine tramite applicazione di una matrice kernel all'immagine originale; L'operazione consiste nella sovrapposizione della matrice kernel alla più grande matrice-immagine procedendo con la moltiplicazione dei pesi del filtro con i pixel dell'immagine fino a quando tutta la matrice-immagine viene completata.
- di *Pooling* che permettono di desumere le informazioni più importanti di una certa area dell'immagine applicando una matrice di una dimensione prestabilita alla matrice-

immagine; solitamente le informazioni prelevate sono il valore massimo (MaxPooling) o medio (AveragePooling).

- non lineari (ad esempio *ReLU*) che permettono di approssimare la funzione che la rete neurale tenta di apprendere, modificando i valori forniti in input dallo strato precedente applicandovi una funzione.
- di elaborazione del risultato generato dalle operazioni precedenti con strati *fully-connected*, per poi fornire l'esito della classificaizone in output.

2.1 Struttura

La rete che è stata implementata risulta così composta:

- **Conv2D - 1**: 16 neuroni (dimensione del banco), dimensione dell'input 28x28x1 (dimensioni dell'immagine, un solo canale per rappresentare il colore).
  In questo strato sono generati 416 parametri a partire da una matrice filtro di dimenisione 5x5, viene applicato zero padding, l'output ha forma 28x28x16.
- **Funzione di attivazione ReLU**: non genera alcun parametro.
- **MaxPooling - 1**: Dato un input di dimensione 28x28x16 applica una matrice kernel di dimensione 2x2, generando un output grande 14x14x16.
  Non genera parametri.
- **Conv2D - 2**: 16 neuroni (dimensione del banco), dimensione dell'input 14x14x16.
  In questo strato sono generati 4112 parametri a partire da una matrice filtro di dimenisione 4x4, non viene applicato zero padding, l'output ha forma 11x11x16.
- **Funzione di attivazione ReLU**: non genera alcun parametro.
- **MaxPooling - 2**: Dato un input di dimensione 11x11x16 applica una matrice kernel di dimensione 3x3, generando un output grande 3x3x16.
  Non genera parametri.
- **Flatten**: Si occupa di appiattire la struttura-tensore finora utilizzata in un vettore di dimensione 144 (3*3*16).
- **Strato Dropout - 1**: il 20% di unità sono spente casualmente in fase di training.
- **Strato Dense - 1**: 16 neuroni completamente connessi allo strato superiore, genera 2320 parametri.
- **Funzione di attivazione ReLU**: non genera alcun parametro.
- **Strato Dropout - 2**: il 20% di unità sono spente casualmente in fase di training.
- **Strato di Output Dense - 2**: 10 neuroni completamente connessi allo strato superiore, genera 170 parametri.
  La funzione di attivazione associata ad esso è di tipo *Softmax*, permette di determinare la classe associata all'osservazione come risultato.

La rete è stata implementata secondo il seguente schema:
Fornita l'immagine in input si procede ad effettuare una operazione di convoluzione tramite la matrice kernel di dimensione 5x5 e zero padding, ottenendo la stessa dimensione in output ma con un immagine modificata dall'applicazione dei pesi della matrice filtro; successivamente si procede ad applicare la funzione di attivazione ReLU sui risultati prima di procedere allo strato di MaxPooling - 1 che dimezza la dimensione dell'input tramite una matrice kernel di dimensione 2x2, la quale riassume le informazioni contenute nell'area considerata dalla sovrapposizione restituendo il valore con intensità maggiore;
Si procede poi a ripetere la medesima operazione di convoluzione e pooling sul risultato con delle matrici kernel differenti (rispettivamente 4x4 per lo strato di Conv2d - 2 e 3x3 per lo strato di MaxPooling - 2).
Conclusa la fase di convoluzione e pooling si procede ad appiattire un risultato già ridotto alla dimensione 3x3x16 tramite una operazione di Flatten, procedendo poi a fornire il risultato ad un layer Dropout - 1 che permette di rendere la rete più robusta rispetto al fenomeno dell'overfitting.
Si inserisce quindi un layer di tipo Dense (con funzione di attivazione ReLU) per elaborare tramite 16 neuroni e relativi pesi una rappresentazione dei dati che verrà poi fornita in output ad un nuovo

strato Dropout - 2 identico al precedente; Per ultimo si considera lo strato Dense - 2 con dimensione 10 neuroni e funzione di attivazione Softmax, utilizzato per la classificazione della rappresentazione tramite il vettore One-Hot.

Si riassume il numero di parametri per ogni Layer:

- Conv2D - 1: 416 parametri.
- MaxPooling - 1: Non genera parametri.
- Conv2D - 2: 4112 parametri.
- MaxPooling - 2: Non genera parametri.
- Flatten: Non genera parametri.
- Strato Dropout - 1: Non genera parametri.
- Strato Dense - 1: 2320 parametri.
- Strato Dropout - 2: Non genera parametri.
- Strato di Output Dense - 2: 170 parametri.

Il numero totale di parametri generati è 7018.

2.2 Analisi degli Iperparametri

2.2.1 Batch Size

A seguito di numerose combinazioni di parametri e valutazioni, la dimensione ottimale del batch è **64**. Risulta essere un numero molto contenuto di osservazioni da passare al modello per essere elaborate prima di aggiornare i pesi della rete.

2.2.2 Epoche

Sono state scelte **50** epoche, il valore è stato mantenuto tale sin dalla prima versione del modello in quanto è stato osservato che già con un valore inferiore di epoche il modello non incrementa ulteriormente le performance generando un plateau.
Saranno effettuate poi delle considerazioni sul numero di epoche riguardo all'early stopping.

2.2.3 Loss Function

La Loss Function scelta è la **Categorical Crossentropy**, una funzione di perdita consigliata dalla letteratura per i casi di classificazione non binaria. È stata scelta poichè si utilizza nei casi in cui solo un possibile esito-categoria si può associare a ciascuna osservazione: una immagine può corrispondere ad una sola lettera, non a più di una.
Questa funzione è stata scelta inoltre perchè lavora correttamente con i risultati forniti dalla funzione di attivazione dell'ultimo layer di Output *Softmax* (che restituisce una probabilità per ogni classe a cui l'osservazione può appartenere), quando vi si fornisce un vettore predetto in codifica one-hot essa calcola una misura di quanto differisce dalla codifica one-hot del vettore reale; più sarà diverso il valore della codifica, maggiore sarà la Loss.

2.2.4 Optimizer

È stato scelto **Adam**, una combinazione di RMSProp e di AdaGrad con momentum: permette rispetto al metodo SGD di ridurre i tempi di calcolo per raggiungere l'ottimo, anche se non sempre è migliore rispetto a quello fornito dal più lento SGD. Il vantaggio che si trae dal suo utilizzo è legato alla sua capacità di adattare il "*learning rate*", cioè un iperparametro che permette di controllare, in funzione dell'errore commesso, di quanto modificare i pesi della rete ogni volta che avviene un aggiornamento degli stessi. Inizialmente l'ottimizzatore della rete era SGD, successivamente durante la fase di test dei vari modelli è risultato più performante Adam e quindi SGD è stato sostituito.

2.2.5 Early Stopping

È stato implementato il meccanismo di **Early Stopping**, una funzione di *Callback* che rimane in ascolto dei risultati che il modello sta garantendo in fase di Training e Validation per decretare uno stop dell'apprendimento dei parametri quando la Validation Loss incrementa per un numero di volte superiore a quelle indicate dalla *Patience* (pari a 7) o nel caso in cui la Validation Accuracy decrementi per un numero di volte superiore a 7: il modello quindi salva gli ultimi parametri che ha appreso e la fase di training si conclude.
È un meccanismo di regolarizzazione che permette di evitare il fenomeno di overfitting.

3 Performance

Il modello ha ottenuto le seguenti performance in fase di Training su 50 epoche:

- un valore di **Loss** pari a 0.078 nel training, 0,044 nella fase di validation. I due valori sono molto vicini, graficamente permettono di

osservare che non si verifica una situazione di overfitting.

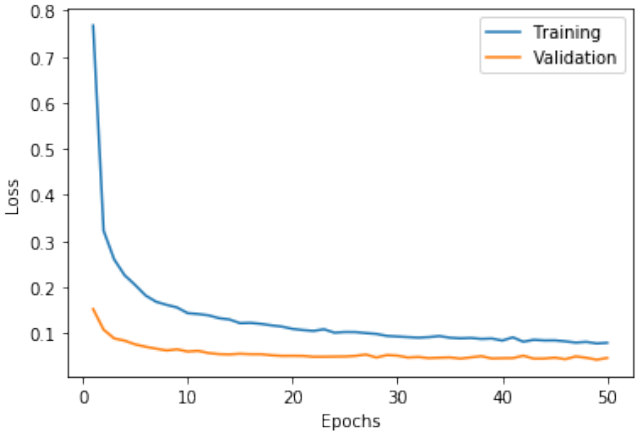

Figura 2: Loss per Training e Validation su 50 epoche

- un valore di **Accuracy** pari a 97.56% nel training, 98.63% nella fase di validation.

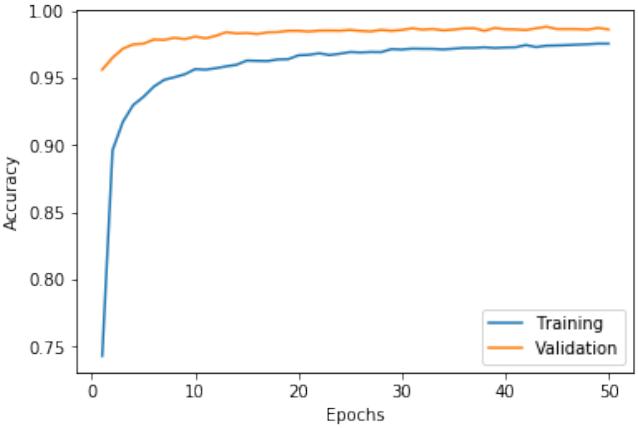

Figura 3: Accuracy per Training e Validation su 50 epoche

- un valore di **F-measure** pari a 97.62% nel training, 98.66% nella fase di validation.

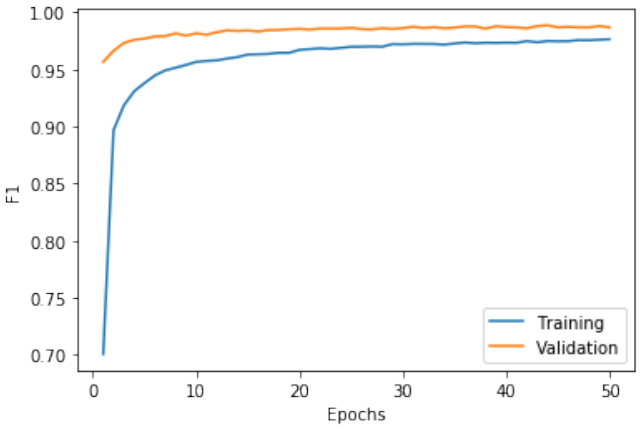

Figura 4: F-measure per Training e Validation su 50 epoche

Per effettuare la fase di Test del modello è stata utilizzata la funzione *predict* sui dati di Test (x_test, y_test), ed il risultato è stato valutato tramite la funzione *classification_report* che ha generato la seguente tabella in cui sono mostrati i valori di Recall, Precision e F1 Measure:

	precision	recall	f1-score	support
0	0.99	0.99	0.99	985
1	0.99	0.99	0.99	1129
2	0.99	0.98	0.99	1042
3	1.00	0.99	0.99	1019
4	0.98	0.99	0.99	971
5	0.98	0.98	0.98	893
6	0.99	0.99	0.99	957
7	0.99	0.99	0.99	1029
8	0.98	0.99	0.99	966
9	0.98	0.98	0.98	1009
micro avg	0.99	0.99	0.99	10000
macro avg	0.99	0.99	0.99	10000
weighted avg	0.99	0.99	0.99	10000
samples avg	0.99	0.99	0.99	10000

Figura 5: Valore delle metriche in fase di Test

Per ottenere i risultati di Loss e Accuracy è stata applicata la funzione *evaluate* sui dati di test, producendo:

- **Loss**: 0.0385
- **Accuracy**: 98.8%