Assignment III

1920-2-F1801Q151, Advance Machine Learning, 2019-2020

Università degli studi di Milano Bicocca
Dipartimento di Informatica, Sistemistica e Comunicazione

Villa Giacomo `807462`

11 Novembre 2019

Indice

1 Dataset e Operazioni preliminari

Per l'assignment è stato richiesto di implementare un classificatore (CNN) al fine **riconoscere** i records del dataset `mnist` i quali rappresentano **numeri scritti a mano**:

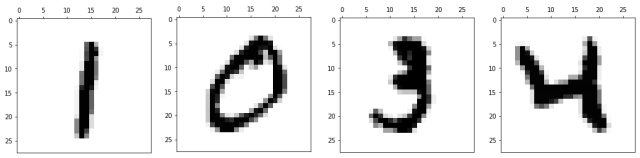

Figura 1: Esempio record da classificare

Dunque l'**immagine** è di fatto una matrice `28x28x1` **ogni cella** detiene un valore numerico che di fatto rappresenta l'**intensità luminosa del pixel** in questione (range `[0, 255]`). Al fine di permettere un corretto funzionamento è stato effettuato un **preprocessing** dove si è provveduto a **riscalare i valori** delle celle in un range `[0, 1]`; è stato inoltre **cambiata la rappresentazione dell'output** non più un valore nel range [0, 9] ma un one-hot vector con 10 elementi, 9 dei quali saranno `0` ed il restant sarà `1`, in una posizione ad indicare il valore di partenza. Il **train set** è composto da **60000 records**, il **test set** è invece composto da **10000 records**.

È stato inoltre effettuato uno studio al fine di **identificare eventuali sbilanciamenti nel dataset** di training, si è poi provveduto ad estrapolare il validation set ripentendo l'analisi; i risultati sono i seguenti:

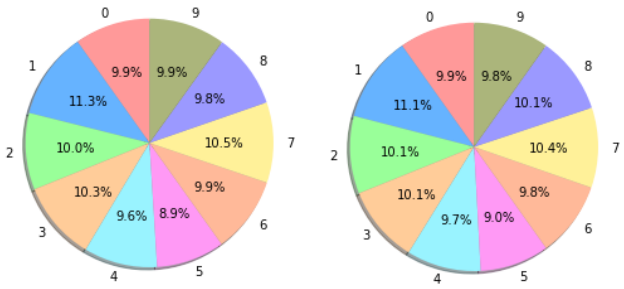

Figura 2: Distribuzione classi nel training set e nel validation set

Risulta possibile osservare come **non vi siano problemi di sbilanciamento** e di come la rappresentanza delle classi, dato il validation set, risulti rispettare le percentuali del dataset originale.

2 Struttura CNN

Nell'assignment veniva richiesto che il **numero di parametri** per questa rete rimanesse sotto le 7500 unità; a tal proposito, per la prima parte della rete, si è **reso fondamentale l'utilizzo** di layer `MaxPooling2D` utili per operazioni di riduzione. La mia rete ha la seguente struttura:

Layer (type)	Output Shape	Param #
conv2d_7 (Conv2D)	(None, 24, 24, 10)	260
max_pooling2d_7 (MaxPooling2	(None, 8, 8, 10)	0
conv2d_8 (Conv2D)	(None, 6, 6, 7)	637
max_pooling2d_8 (MaxPooling2	(None, 3, 3, 7)	0
dropout_10 (Dropout)	(None, 3, 3, 7)	0
flatten_4 (Flatten)	(None, 63)	0
dense_9 (Dense)	(None, 64)	4096
dropout_11 (Dropout)	(None, 64)	0
dense_10 (Dense)	(None, 32)	2080
dropout_12 (Dropout)	(None, 32)	0
dense_11 (Dense)	(None, 10)	330

Total params: 7,403
Trainable params: 7,403
Non-trainable params: 0

Figura 3: Struttura Rete

Il **numero di parametri** utilizzato risulta essere **minore rispetto al vincolo** imposto, in particolare la rete è la seguente:

```
def define_model():
    model = Sequential()
    model.add(Conv2D(10, kernel_size=(5, 5), activation='relu',
        input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(3, 3)))
    model.add(Conv2D(7, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.3))
    model.add(Flatten())
    model.add(Dense(64, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(32, activation='relu'))
    model.add(Dropout(0.2))
```

```
14      model.add(Dense(10, activation='softmax'))
15      model.compile(loss='categorical_crossentropy',
16          optimizer = "Adadelta", metrics=['accuracy'])
17
18      model.summary()
19      return model
```

Ho deciso di implementare due livelli di `MaxPooling2D` al fine di poter **diminuire** in seguito il **numero di parametri** necessari alla rete. Ho definito **due banchi di filtri** di, rispettivamente, 10 e 7; questo in quanto un **buon numero di filtri** mi permette di **aumentare** il numero di **features che la rete sarà in grado di apprendere**. Ho implementato **due layers densi** di 64 e 32 al fine di poter utilizzare i parametri rimasti fornendo una **struttura a "triangolo"** alla rete; l'**ultimo layer** è rappresentato dai 10 possibili **output**.

I **layer di dropout** presenti permettono di minimizzare la possibilità di cadere in situazioni di overfitting.

3 Iperparametri della Rete

Per quanto riguarda la **funzione di attivazione** dei **layers interni** ho deciso di utilizzare la **ReLu**, scelta giustificata dallo stato dell'arte e dal fatto che questa permette di ottenere un apprendimento relativamente più rapido, grazie alla sua natura, rispetto ad altre funzione di attivazione.

Come **funzione di attivazione del layer di output** ho deciso di utilizzare una `Softmax` in quanto, il problema in questione, è una **classificazione multiclasse**; di fatto la funzione `Softmax` mi permette di ottenere un vettore che può essere trattato come un vettore di probabilità, che dunque mi permetterà di adottare la `Categorical Crossentropy` come **funzione di loss** al fine di minimizzare l'entropia tra ciò che dovrebbe essere (nella versione one-hot vector) e ciò che effettivamente il mio modello predice.

La **dimensione dei batch** è ricaduta su **256**, il valore mi sembra abbastanza corretto per far si che la discesa mediante gradiente segui un buon andamento. Ho **effettuato test** con differenti valori e non sembra cambiare le performance, reputo comunque corretto mantenere la dimensione definita.

Il **numero di epoche** è stato impostato a **200**; tuttavia **tale valore non viene mai effettivamente raggiunto**; viene di fatto impostato un `Early Stopping` al fine di **fermare la fase di fitting** nel momento in cui la loss (sul validation set), per un periodo di 5 epoche consecutive, non registri un miglioramento; tale scelta mi ha portato ad ottenere una **fase di training lunga 52 epoche**.

Per quanto riguarda l'**ottimizzatore** la scelta è ricaduta su **Adadelta**; ho comunque effettuato diversi test con diversi differenti ottimizzatori, la scelta è ricaduta su questo in quanto **reputo più corretto il fatto che adatti il learning rate** in funzione degli update del gradiente data, però, una finestra. Il **valore iniziale** del **learning rate** è pari a **1.0** e il **fattore di decadenza** pari a **0.95**, **non sono stati cambiati** in quanto la documentazione consigliava di lasciare inalterati i parametri.

4 Performance

Nella prima fase di **allenamento della rete** è stato effettuato uno **splitting del training set** in un train set effettivo e in un validation (splitting 75% - 25%) al fine di osservare le **curve** per quanto riguarda **Accuracy** e **Loss**. Le curve stesse risultano essere le seguenti:

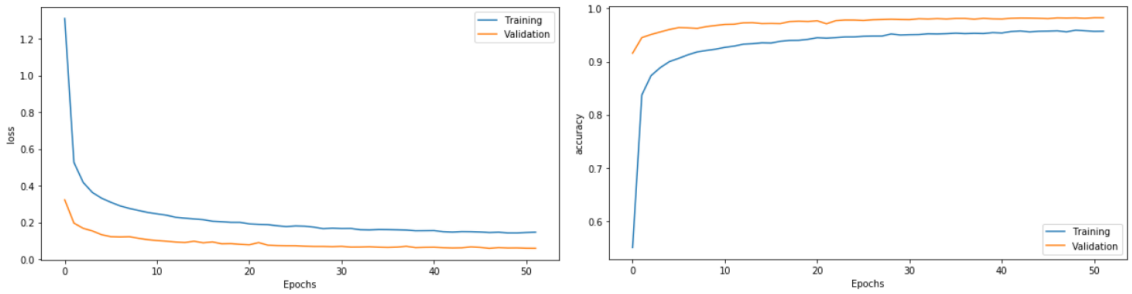

Figura 4: Curve di Loss e Accuracy

Possiamo osservare come le **curve del validation set** (sia per quanto riguarda la Loss che l'Accuracy) **risultano essere sotto (o sopra) quelle di training**; questo risultato è dovuto ai vari layer di `Dropout` volti a diminuire l'adattamento del **modello** rendendolo, di fatto, **in grado di generalizzare meglio**. Il valore di **Accuracy** finale sul **Validation** è pari a `0.9817` e il valore della **Loss** è pari a `0.0640`.

Le **performance** sono state poi misurate sul `testset` messo a disposizione dal download del dataset stesso data la libreria `Keras`; lo stesso è **composto da 10000**

records (con la relativa true table) in formato analogo a quanto già mostrato nella sezione `1 Dataset e Operazioni preliminari` di questa documentazione. Dunque i **risultati sono i seguenti**:

Performance				
	Precision	**Recall**	**f1-score**	**support**
0	*0.98*	*0.99*	*0.99*	*980*
1	*0.99*	*0.99*	*0.99*	*1135*
2	*0.98*	*0.99*	*0.98*	*1032*
3	*0.99*	*0.99*	*0.99*	*1010*
4	*1.00*	*0.98*	*0.99*	*982*
5	*0.98*	*0.99*	*0.99*	*892*
6	*1.00*	*0.98*	*0.99*	*958*
7	*0.98*	*0.98*	*0.98*	*1028*
8	*0.98*	*0.99*	*0.98*	*974*
9	*0.98*	*0.97*	*0.98*	*1009*
micro avg	*0.98*	*0.98*	*0.98*	*10000*
macro avg	*0.98*	*0.98*	*0.98*	*10000*
weighted avg	*0.98*	*0.98*	*0.98*	*10000*

Possiamo notare come le **performance siano ottime e l'errore sia basso**, l'**Accuracy** totale si attesta intorno al *0.9844* con un **valore di Loss** pari a *0.05*. Fornisco anche la matrice di confusione a conferma dei risultati sovrariportati:

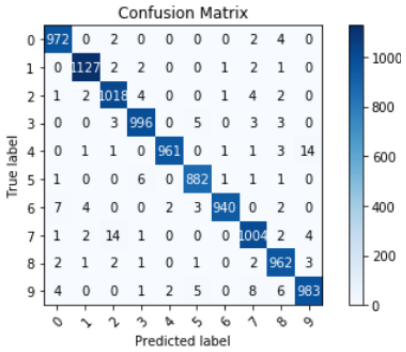

Figura 5: Matrice di confusione dato il testset

Unica problematica degna di nota potrebbe essere quella che vede etichettare record della classe **7** con la label della classe **2** e quelli della classe **4** con la label della classe **9**. In generale non sono presenti grosse problematiche.

Ho infine analizzato una **approssimazione in due dimensioni** della **capacità discriminativa** del modello, nel layer **subito dopo** l'area **CNN** e **subito prima dell'output**. Ho voluto verificare se la capacità discriminativa migliori tra l'ingresso nella zona dei layers densi e l'uscita da questa verso l'output. I risultati da me ottenuti sono i seguenti:

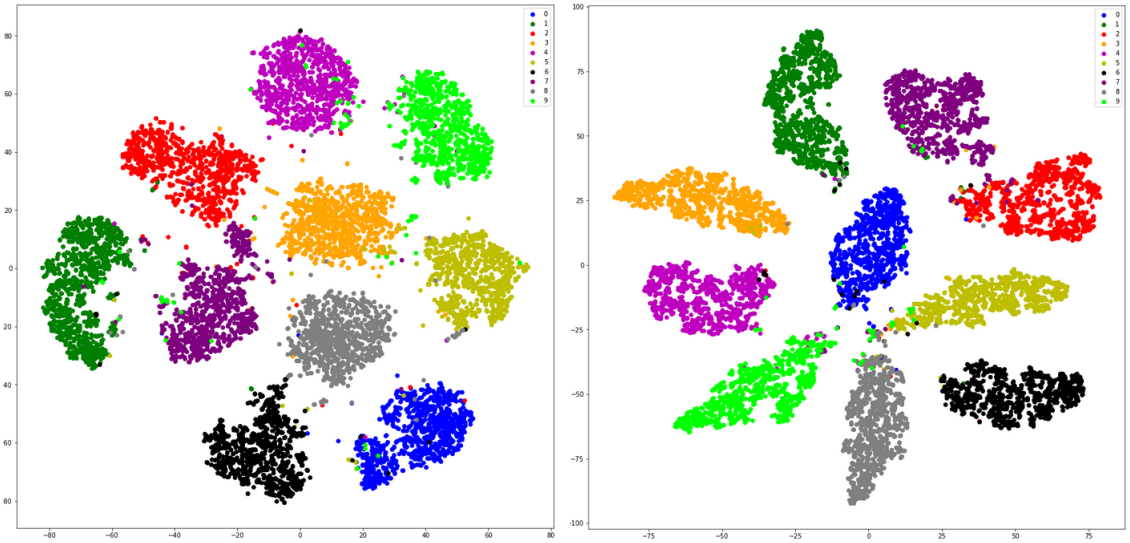

Figura 6: Capacità discriminativa dopo CNN e prima dell'output

Possiamo osservare come, **dopo i due layer densi**, la rappresentazione grafica degli elementi delle varie classi abbia subito un **"ricompattamento"**, in generale è chiaro che la **capacità discriminativa ha visto un aumento**.

Gli **errori totali**, sul testset, sono **155** su un totale di 10000 elementi (dunque un *1,55%* sul totale dei record). Fornisco alcuni errori, nel notebook sono presenti tutti e 155:

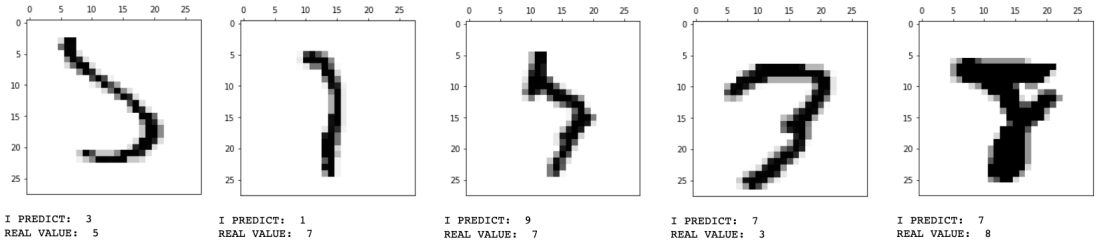

Figura 7: Esempio errori commessi dalla rete

Possiamo osservare come questi possano essere **difficilmente identificabili** anche da un umano. **Esempi di validi output** possono essere i seguenti:

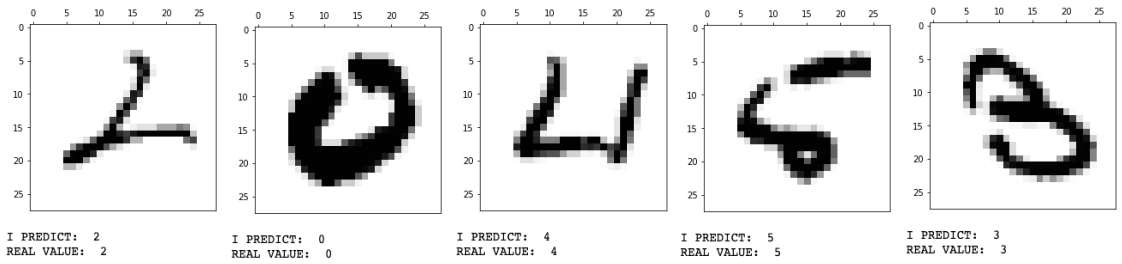

Figura 8: Esempio inferenze corrette della rete

Anche questi sono presenti nel notebook e dunque visualizzabili.

5 Conclusioni

In generale i **risultati ottenuti sono alquanto soddisfacenti** dato il compito assegnato, **non si presentano particolari problematiche** di inferenza dato il modello.

Sarebbe interessante provare ad **aumentare il numero dei parametri** osservando i risultati al fine di verificare eventuali miglioramenti degli stessi; sarebbe inoltre interessante un **confronto con una rete neurale "classica"** dotata dello **stesso numero di attributi** al fine di confrontare i risultati ottenuti.