

Advanced Machine Learning: Assignment #3

Fabrizio D'Intinosante — 838866

Università degli Studi di Milano Bicocca — November 3, 2019

Introduzione

L'obiettivo dell'*assignment* è quello di creare una CNN per classificare le istanze del conosciuto *dataset* **MNIST**, composto da 70.000 immagini 28x28 *gray-scale* rappresentanti i numeri da 0 a 9 con l'unica restrizione che il numero totale di parametri da stimare resti sotto il tetto massimo di 7500. Il *dataset* acquisito tramite *keras* si presenta in porzioni 60.000 *training set* e 10.000 *test set*.

1 Esplorazione e preprocessing

Come primo passaggio si è provveduto ad ispezionare il *dataset* per verificare graficamente la numerosità delle diverse classi, trovando una composizione piuttosto bilanciata; successivamente si è proceduto a rappresentare graficamente alcune istanze come si vede in fig. 1.

Come primo elemento di *preprocessing* si è effettuato un ulteriore partizionamento del *training set* in *validation set* con una proporzione 90%-10% per poi procedere a riscalare il valore dei pixel delle immagini dall'originale intervallo $[0, 255]$ a quello $[0, 1]$.



NB L'operazione di *rescaling* è stata effettuata su tutte le partizioni presenti: *training*, *validation* e *test set*.

Infine si è provveduto a convertire la classe etichetta delle immagini in formato *one-hot encoding* così da permettere il processo di classificazione.

2 Modello

Il modello realizzato presenta un'architettura piuttosto basilare; ciò è dovuto alla relativa semplicità rappresentata dalla classificazione del *dataset* **MNIST**.

Come si può vedere in fig. 2 la rete è composta essenzialmente da due *layers convolutionals*, il primo composto da 10 filtri 3x3 mentre il secondo da 4 filtri sempre 3x3, entrambi eseguiti con una strategia *zero-padding* e seguiti ognuno da un *layer* di *max pooling* (nello specifico con una dimensione della finestra 2x2, uno *strides* di 2 ed un *padding* di *default*), ed uno di *batch normalization*, una pratica piuttosto standard. Successivamente è presente un *layer flatten*, necessario per rendere *flat* le immagini così da permettere il collegamento con *layers fully connected*, necessari al processo di classificazione. Successivo al *layer fully connected* composto da 32 neuroni è presente un *dropout layer* per evitare *overfitting* con un rate di 0,1. La rete infine termina con un *output layer* con un numero di neuroni uguale al numero di classi presenti nei nostri dati, ovvero 10. Tutti i *layers* utilizzati come funzione di attivazione utilizzano una ReLU, una



Figure 1: Esempio di istanze *dataset* MNIST

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 28, 28, 1)	0
conv2d_1 (Conv2D)	(None, 28, 28, 10)	100
max_pooling2d_1 (MaxPooling2)	(None, 14, 14, 10)	0
batch_normalization_1 (Batch Normalization)	(None, 14, 14, 10)	40
conv2d_2 (Conv2D)	(None, 14, 14, 4)	364
max_pooling2d_2 (MaxPooling2)	(None, 7, 7, 4)	0
batch_normalization_2 (Batch Normalization)	(None, 7, 7, 4)	16
flatten_1 (Flatten)	(None, 196)	0
dense_1 (Dense)	(None, 32)	6304
dropout_1 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 10)	330
Total params: 7,154		
Trainable params: 7,126		
Non-trainable params: 28		

Figure 2: *Summary della rete utilizzata*

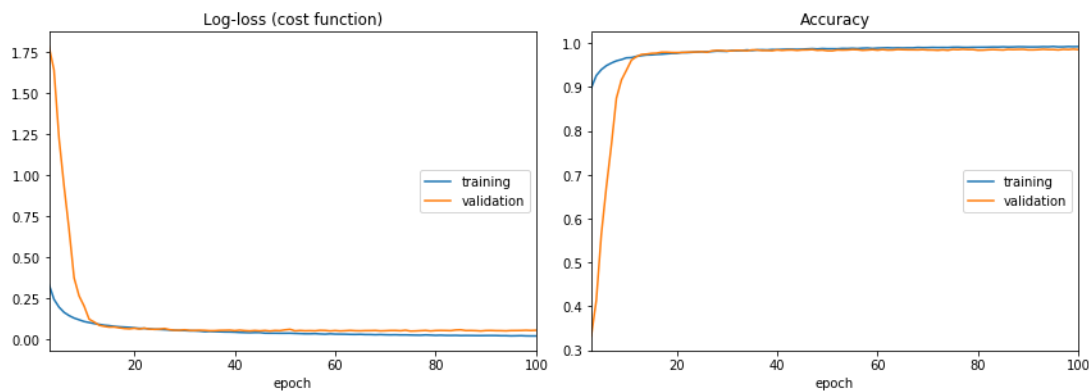


Figure 3: *Plots raffiguranti loss e accuracy per validation e train set*

scelta piuttosto standard che si è dimostrata ottimale in questo caso, fatta eccezione per l'*output layer* che utilizza una *softmax*, necessaria per la classificazione multiclasse.

Come si può vedere sempre in fig. 2 il numero di parametri per *layer* è sostanzialmente ridotto nella parte "convoluzionale" della rete mentre, come prevedibile, esplode al momento della conversione dell'immagine e del collegamento di ogni pixel con i neuroni del primo *layer fully connected*. Il numero totale di parametri si attesta a 7154, al di sotto della soglia massima fissata.

Gli altri iper-parametri utilizzati per il *training* della rete sono:

- *batch size* di 1024, computazionalmente più onerosa ma più rapida nell'esecuzione complessiva;
- ottimizzatore adam, molto potente rientrando nel gruppo degli ottimizzatori adattivi;
- 100 epoche si sono dimostrate più che sufficienti ad ottenere un ottimo risultato come si può vedere in fig. 3, evitando inoltre di cadere in *overfitting*;
- *loss function categorical crossentropy*, una scelta obbligata data la natura multiclasse della classificazione.

3 Conclusioni

In conclusione, si può affermare che la rete realizzata raggiunge risultati molto buoni su tutte e tre le partizioni di dati, come si può vedere in fig. 4 ed in fig. 5. In particolare in fig. 5 è possibile distinguere quali classi siano classificate meglio rispetto alle altre: anche se si tratta di differenze molto piccole e probabilmente non statisticamente significative, il valore di *f1-score* si rivela più alto per le immagini rappresentanti 0 e 1, mentre tende ad essere leggermente più basso per i numeri da 2 a 9.

```

Log-loss (cost function):
training (min: 0.022, max: 1.592, cur: 0.022)
validation (min: 0.052, max: 1.794, cur: 0.057)

Accuracy:
training (min: 0.481, max: 0.993, cur: 0.992)
validation (min: 0.331, max: 0.987, cur: 0.986)

```

Figure 4: Min, Max e valore finale di *loss* e *accuracy* su *train* e *validation set*

	precision	recall	f1-score	support
0	0.9888	0.9949	0.9919	980
1	0.9965	0.9921	0.9943	1135
2	0.9893	0.9874	0.9884	1032
3	0.9814	0.9911	0.9862	1010
4	0.9888	0.9878	0.9883	982
5	0.9778	0.9877	0.9827	892
6	0.9894	0.9739	0.9816	958
7	0.9893	0.9883	0.9888	1028
8	0.9767	0.9897	0.9832	974
9	0.9879	0.9742	0.9810	1009
accuracy			0.9868	10000
macro avg	0.9866	0.9867	0.9866	10000
weighted avg	0.9868	0.9868	0.9868	10000

Figure 5: *Classification report* del modello sul *test set*