Week 3 — Assignment Submission

Gianluca Scarpellini - 807541 - g.scarpellini1[at]disco.unimib.it
(In team con Federico Belotti - 808708)

10 novembre 2019

Indice

1 Task

Il task oggetto di questo assignment riguarda l'addestramento di una rete neurale convoluzionale (CNN) per la classificazione di numeri scritti a mano. La consegna prevedeva inoltre una costrizione sul numero di parametri della rete, che non doveva eccedere i 7,500. Il dataset addotato per il training e la validazione dei modelli è il famoso MNIST [LC10]. Nella sezione 2 vengono riportate le caratteristiche del dataset e il preprocessing che abbiamo adottato. Negli esperimenti effettuati abbiamo analizzato diverse possibili architetture con un numero di parametri dai 700 ai 7,500. I due modelli finali, che presento alla sezione 3, constano rispettivamente di 7,498 e 722 parametri e raggiungono performance da stato dell'arte. Nella sezione 4 riporto gli esperimenti effettuati e i risultati raggiunti sul testset di Mnist. Infine, in 5 analizzo possibili sviluppi futuri nell'ambito DL e in particolare nello sviluppo di CNN per task di classificazione con relativmente pochi parametri.

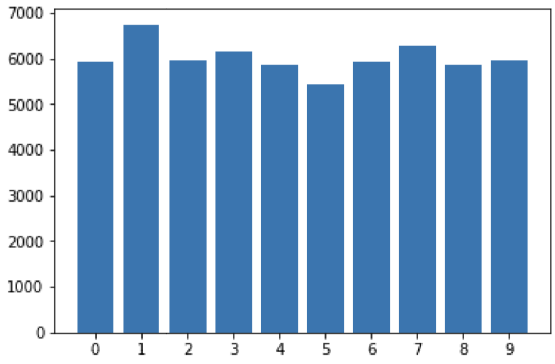


Figura 1: Distribuzione delle cifre del dataset MNIST

2 Dataset e preprocessing

Il dataset impiegato per l'addestramento è il Mnist . Il dataset consta di 60,000 immagini in scala di grigio, di cui 48,000 per il training e 12,000 per il test. Abbiamo deciso di estrarre dal training set un subset di immagini (che chiameremo val set) con cui valutare la bontà della scelta degli ipermateri. Non abbiamo effettuato preprocessing, salvo riportare i valori di intensità nel range double [0, 1]. Abbiamo deciso di non effettuare data augmentation in quanto il dataset non presenta una varianza signficativa: le cifre sono state acquisite su sfondo nero con proporzioni e luminosità tra loro molto simili. Abbiamo pertanto optato per una normalizzazione del dataset, sottraendo per media e dividendo per deviazione standard dello stesso al fine di centrare il dataset in zero con varianza 1. In 1 riporto l'istogramma della distribuzione del dataset per classe. Questo risulta essere ben bilanciato.

3 Modello

3.1 CNN con 722 parametri

Il primo modello che riporto è una CNN con 3 soli layer convluzionali. Per ciascun layer abbiamo optato per un kernel size di (3x3), con stride 2. In questo modo il nostro modello riduce la dimensione spaziale dell'input (inizialmente di 28x28) aumentandone al contempo la profondità (numero di canali). Abbiamo mentenuto contenuto il numoer di canali di profondità, in modo che al momento del flatten ciascun immagine fosse rappresentata da un array di soli **32 elementi**.

3.2 CNN con 7,498 parametri

Il secondo modello è una CNN che segue l'approccio inizialmente introdotto da [KSH12], ossia una sequenza di $CNV \rightarrow RELU \rightarrow Pool$ che incrementalmen-

Layer (type)	Output Shape	Param #
conv2d_26 (Conv2D)	(None, 13, 13, 2)	20
activation_25 (Activation)	(None, 13, 13, 2)	0
conv2d_27 (Conv2D)	(None, 6, 6, 4)	76
activation_26 (Activation)	(None, 6, 6, 4)	0
conv2d_28 (Conv2D)	(None, 2, 2, 8)	296
activation_27 (Activation)	(None, 2, 2, 8)	0
flatten_8 (Flatten)	(None, 32)	0
dense_8 (Dense)	(None, 10)	330

Total params: 722
Trainable params: 722
Non-trainable params: 0

Figura 2: Modello impiegato (722 parametri)

te aumentino la depth del tensore e ne riducono la dimensione spaziale. Nel nostro approccio abbiamo deciso di impiegare diversi layer di normalizzazione (BatchNorm). Difatti, nonostante la normalizzazione del training set, la normalizzazione è spesso citata in letteratura per il boost del tempo del training e per le sue peculiarità di regolarizzazione del training [GBC16; IS15]. Abbiamo inoltre valutato le performance nel caso in cui un layer di Dropout, anch'esso con funzioni di regolarizzazione, fosse introdotto in testa alla coppia di layer Fully Connected.

Layer (type)	Output Shape	Param #
conv2d_111 (Conv2D)	(None, 26, 26, 8)	80
activation_111 (Activation)	(None, 26, 26, 8)	0
max_pooling2d_110 (MaxPoolin	(None, 13, 13, 8)	0
batch_normalization_22 (Batc	(None, 13, 13, 8)	32
conv2d_112 (Conv2D)	(None, 11, 11, 16)	1168
activation_112 (Activation)	(None, 11, 11, 16)	0
max_pooling2d_111 (MaxPoolin	(None, 5, 5, 16)	0
batch_normalization_23 (Batc	(None, 5, 5, 16)	64
conv2d_113 (Conv2D)	(None, 3, 3, 32)	4640
activation_113 (Activation)	(None, 3, 3, 32)	0
max_pooling2d_112 (MaxPoolin	(None, 1, 1, 32)	0
batch_normalization_24 (Batc	(None, 1, 1, 32)	128
flatten_38 (Flatten)	(None, 32)	0
dropout_9 (Dropout)	(None, 32)	0
dense_77 (Dense)	(None, 32)	1056
dense_78 (Dense)	(None, 10)	330

Total params: 7,498
Trainable params: 7,386
Non-trainable params: 112

Figura 3: Modello impiegato (7,498 parametri, con Dropout)

4 Esperimenti

Nella fase di esperimentazione abbiamo valutato diversi set di iperparametri, optando infine per i seguenti:

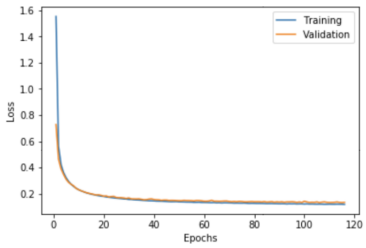
- batch size: 256
- optimizer: Adam con lr di 0.001
- epochs: 100

4.1 Risultati del modello con 722 parametri

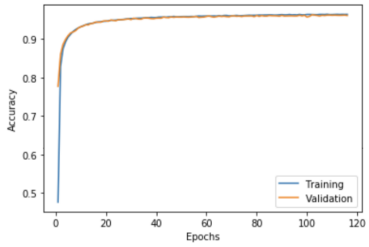
Riporto in figura 4(a) i risultati su training, validation e test ottenuti impiegando il modello descritto in sezione 3.1. Dalle curve di loss e di accuracy nelle figure 4(c) e 4(b) si evince un training del modello costante e privo di overfitting sino all'epoca 100, a riprova della bontà dell'approccio scelto.

Split	**Precision**	**Recall**	**Accuracy**	**F1**
Training	0.96	0.96	0.96	0.96
Validation	0.96	0.95	0.95	0.95
Test	0.96	0.96	0.96	0.96

(a) Tabella dei risultati



(b) Curva di loss

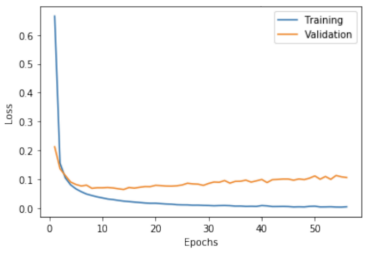


(c) Curva di accuracy

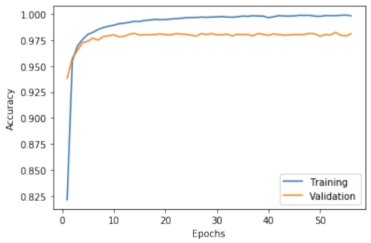
Figura 4: Risultati con il modello di 722 parametri

4.2 Risultati del modello con 7,498 parametri

Riporto in figure i risultati su training, validation e test ottenuti impiegando il modello descritto in sezione 3.2. In particolare, dalle curva di loss in figura 5(a) (e di riflesso anche dalla curva di accuracy) del modello senza dropout si envice un forte overfitting a partire dalla 15esima epoca. L'overfitting è probabilmente dovuto al numero di parametri e alla profondità della rete impiegata che, nonostante i layer di batch normalization, risulta sovra-dimensionata rispetto al task. L'introduzione di un layer di dropout con probabilità 0.3, come descritto in sezione 3.2, determina una maggior regolarizzazione della rete e, conseguentemente, curve di loss e accuracy più smooth (riportate rispettivamente in figura 6(b) e 6(c)).



(a) Curva di loss

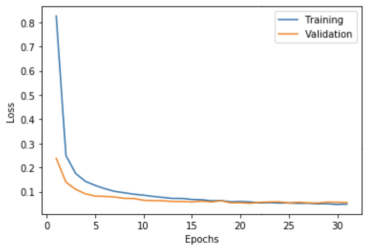


(b) Curva di accuracy

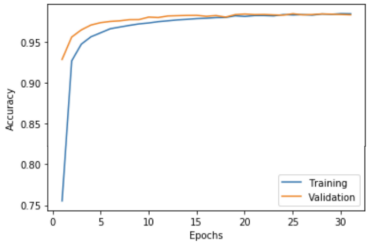
Figura 5: Risultati con il modello di 7,498 parametri (senza dropout)

Split	**Precision**	**Recall**	**Accuracy**	**F1**
Training	1.00	1.00	1.00	1.00
Validation	0.98	0.98	0.98	0.98
Test	0.98	0.98	0.98	0.98

(a) Tabella dei risultati



(b) Curva di loss



(c) Curva di accuracy

Figura 6: Risultati con il modello di 7,498 parametri (con dropout)

5 Conclusione

Il task di classificazione di cifre è spesso citato in letteratura per la sua generalizzabilità e la sua relativa semplicità. In particolare, la letteratura scientifica in ambito ML ha indagato a fondo possibili soluzioni per il dataset MNIST. Nel presente lavoro abbiamo indgato sulla possibilità di impiegare CNN con circa 700 parametri raggiungendo al contempo risultati allo stato dell'arte. Dal confronto delle tabelle riassuntive per gli esperimenti con 722 parametri (in tabella 4(a)) e con 7,498 (in tabella 6(a)) notiamo uno scarto di 2 punti percenutali di quest'ultimo approccio rispetto alla CNN ridotta. Tuttavia, sia in termini di accuracy sia di f-measure (pesate) l'approccio si è attestato sopra il 96%. Possibili sviluppi futuri in questo ambito potrebbero riguardare lo sviluppo di pipeline efficienti nel pruning di reti neurali.

Bibliografia

[LC10] Yann LeCun e Corinna Cortes. "MNIST handwritten digit database". In: (2010). URL: http://yann.lecun.com/exdb/mnist/.

[KSH12] Alex Krizhevsky, Ilya Sutskever e Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: (2012). A cura di F. Pereira et al., pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[IS15] Sergey Ioffe e Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." In: *CoRR* abs/1502.03167 (2015). URL: http://dblp.uni-trier.de/db/journals/corr/corr1502.html#IoffeS15.

[GBC16] Ian Goodfellow, Yoshua Bengio e Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.