Assignment 3
Advanced Machine Learning
University of Milano-Bicocca
M.Sc. Data Science

Alessandro Riboni
November 11, 2019
ID number: 847160
a.riboni2@campus.unimib.it

Design of a CNN Architecture for a Digits Recognition Problem

The task of this assignment is to design a CNN architecture and its training for recognizing handwritten digits. The input dataset used is *MNIST digits*, and it was divided into a training set, with 60000 images 28x28 and a test set, with 10000 images 28x28. The number of classes is 10.

Figure 1: MNIST digits

In this report are presented the following concepts:

- Data Preprocessing;
- Description of the designed architecture;
- Plots of the training and validation loss/accuracy;
- Classification performance on training and test sets.

Data Preprocessing

Before starting to develop a CNN architecture for this problem, it was necessary to make a couple of changes. The images of the dataset have been normalized and reshaped to obtain images define with three values: height, width, and channel. In this situation, there is only one channel because the pictures are in gray scale.
Finally, the class variable has been converted to be suitable for the neural network through a process called *one-hot encoding.*

Description of the designed architecture

In this section there is an overview of the Convolutional Neural Network that was implemented to solve the proposed task:

- an **input layer** with *shape = (28,28,1)*;
- a **convolutional layer** with 16 filters of *kernel_size = 3x3* and activation function *ReLu*;
- a **pooling layer** with Average function and *pool_size = 2x2*;
- a **convolutional layer** with 16 filters of *kernel_size = 3x3* and activation function *ReLu*;
- a **pooling layer** with Average function and *pool_size = 2x2*;
- a **flatten layer** that convert the output shape of the previous layer to a *mono-dimensional array*;
- a **fully-connected output layer** with one neuron for each class (10) and activation function *softmax*.

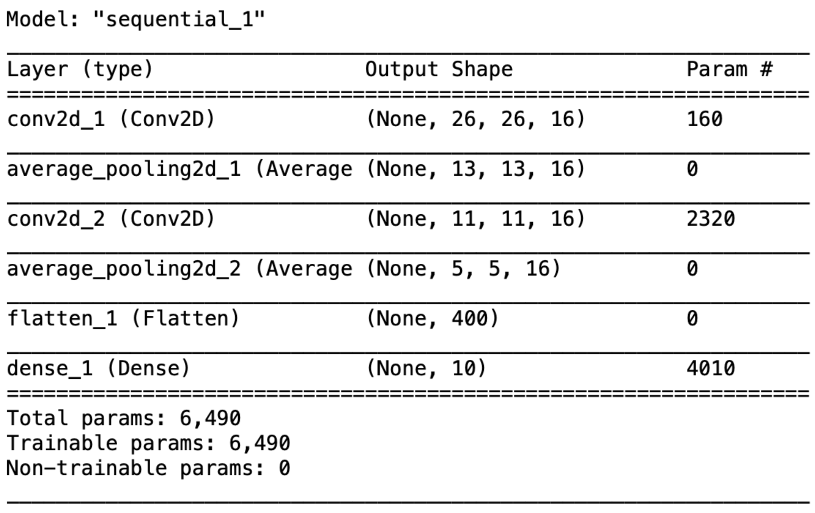

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 26, 26, 16)        160
_________________________________________________________________
average_pooling2d_1 (Average (None, 13, 13, 16)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 11, 11, 16)        2320
_________________________________________________________________
average_pooling2d_2 (Average (None, 5, 5, 16)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 400)               0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                4010
=================================================================
Total params: 6,490
Trainable params: 6,490
Non-trainable params: 0
_________________________________________________________________
```

Figure 2: Model summary

As can be seen from the summary of the network above, the number of parameters is 6490 and it is lower than the maximum limit of 7500 imposed in the assignment. In particular, the number of parameters of the first convolutional layer is equal to the kernel_size plus 1 (bias), multiplied by the number of filters: ((3x3)+1)x16 = 160.
The number of parameters of the second Conv2D layer is ((3x3x16)+1)x16 = 2,320 while, the final Dense layer has (400+1)x10 = 4010 parameters, where 400 is the dimension of the previous flatten layer, 1 is bias and 10 is the number of classes.
The result of using a pooling layer and creating down sampled or pooled feature maps is a summarized version of the features detected in the input.
The excellent results obtained confirm that the relevant information present in the input data is not lost during the pooling phase.
In addition, *Figure 2* shows that the output shape of the first convolutional layer is (26,26,26) because a *no_padding* has been applied. This technique is preferred to *zero-padding* because, being the digits centered, there is no risk of losing information.
The model is compiled using *Adam* optimizer (with *learning_rate = 0.001*) and *categorical cross-entropy* such as loss function. After several tests, the number of epochs have been set to 15 and the batch size equal to 128.
Finally, it was used 10% of the training set as a validation set to control the possible overfitting of the Convolutional Neural Network.

Plots of the training and validation loss/accuracy

In the following figures are presented the accuracy and the loss function on the training set (green) and on the validation set (blue).

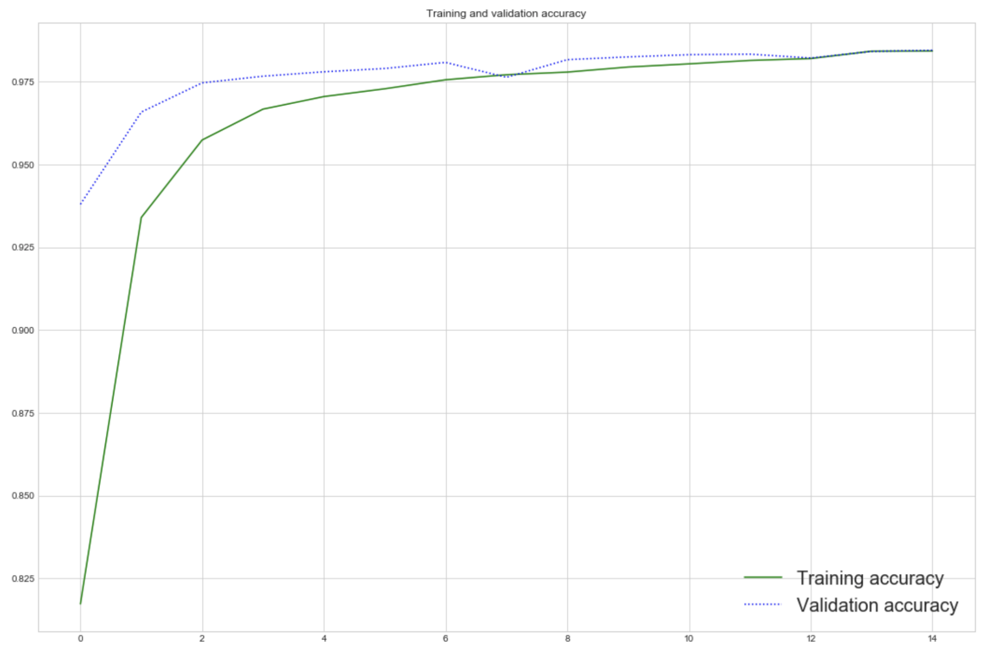


Figure 3: Accuracy on training set and validation set

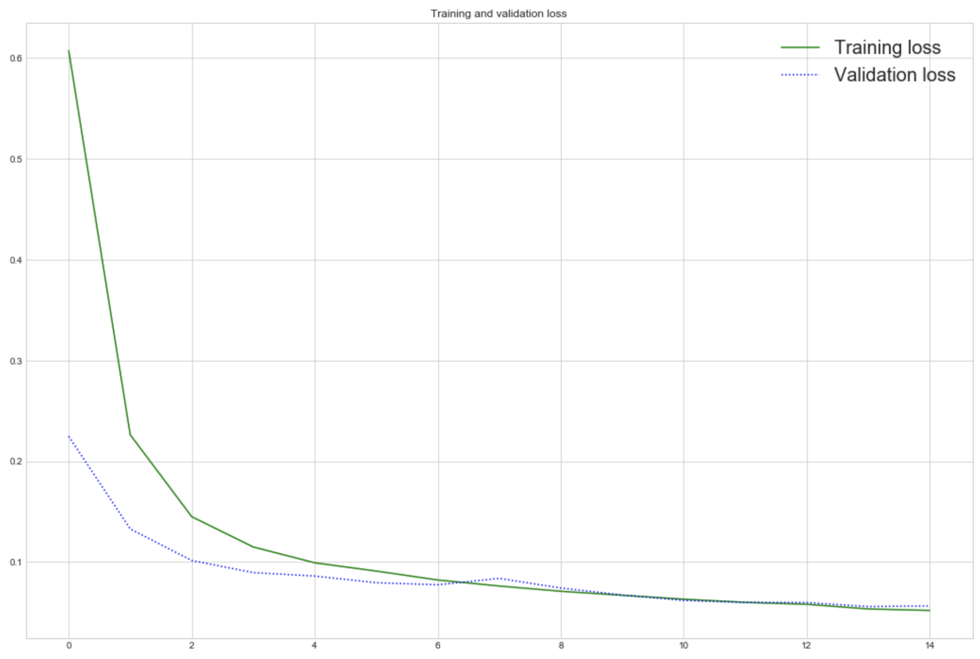
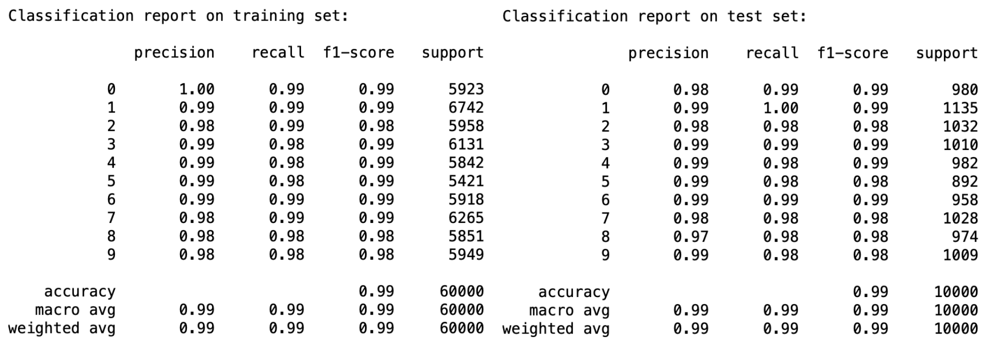


Figure 4: Loss function on training set and validation set

Classification performance on training and test sets

After training the network and avoiding overfitting, the model was evaluated on the data present in the test set. As it can be seen from the following figures, the model continues to obtain excellent results even on data not considered during the training phase. The generalisation error is minimal and the overall accuracy is 99% both on the training and on the test set.



```
Classification report on training set:              Classification report on test set:

              precision    recall  f1-score   support               precision    recall  f1-score   support

           0       1.00      0.99      0.99      5923            0       0.98      0.99      0.99       980
           1       0.99      0.99      0.99      6742            1       0.99      1.00      0.99      1135
           2       0.98      0.99      0.98      5958            2       0.98      0.98      0.98      1032
           3       0.99      0.98      0.99      6131            3       0.99      0.99      0.99      1010
           4       0.99      0.98      0.99      5842            4       0.99      0.98      0.99       982
           5       0.99      0.98      0.99      5421            5       0.99      0.98      0.98       892
           6       0.99      0.99      0.99      5918            6       0.99      0.99      0.99       958
           7       0.98      0.99      0.99      6265            7       0.98      0.98      0.98      1028
           8       0.98      0.98      0.98      5851            8       0.97      0.98      0.98       974
           9       0.98      0.98      0.98      5949            9       0.99      0.98      0.98      1009

    accuracy                           0.99     60000     accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     60000    macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     60000 weighted avg       0.99      0.99      0.99     10000
```

Figure 5: Classification report on training set and test set.