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

Federico Signoretta
ID number: 847343
f.signoretta@campus.unimib.it
November 10, 2019

Image Classification through Convolutional Neural Networks

The task of this assignment is the design of a CNN architecture and its training. The dataset used is the MNIST dataset, downloaded directly from API. The CNN has to be designed with the hard constraint of a maximum of 7.5K parameters.

1 MNIST dataset

The original dataset was already split into training and test. In particular, the database contains 60,000 training images and 10,000 test images, where the size of the images are composed by 28x28 pixel. Those data were reshaped by adding one more dimension: the number of channels (in this case 1). Therefore, in order to normalize the value of the images it was divided by 255 (which is the maximum RGB code minus the minimum RGB code). This pre-processing was necessary in order to implement the CNN by using keras.

Figure 1: Sample images from MNIST dataset

2 Description of the CNN

2.1 Description of the designed architecture

The Convolutional Neural Netwrok implemented is composed by:

- an **input layer** with dimension (28x28x1);
- a **convolutional layer** composed by 16 filters with a sliding window of 3x3 and with activation function "ReLu";
- a **pooling layer** AveragePooling2D with a 2x2 window
- a **convolutional layer** composed by 16 filters with a sliding window of 3x3 and with activation function "ReLu";

- a **pooling layer** AveragePooling2D with a 2x2 window;
- an **output layer** (densely connected layer) with dimension 10 (number of the classes) and with activation function "softmax".

2.2 Parameters count for each layer

The number of parameters of the first conv2D layer corresponds to the weight matrix W of 3x3 and a b bias for each of the filters is 160 parameters (16 (9 + 1)). Average-pooling does not require parameters since it is a mathematical operation to find the average.
The size of the resulting second convolution layer is 11x11 since it starts from an input space of 131316 and a sliding window of 33, taking into account that it has a stride of 1. The number of parameters 2,320 corresponds to the fact that the second layer will have 16 filters with 145 parameters each (1 corresponds to the bias, and a W matrix of 33 for each of the 16 entries). That means ((3x3x16) +1) x 16 = 2,320.
The number of parameters of the Softmax layer is ((10400) + 10), where 400 is the dimension of the flattened vector created before applying the Softmax. That means 4,010 parameters. So, the total amount of parameters is equal to (160 + 2,320 + 4,010) = 6,490 ($\leq$ 7,500 - constraint parameters).

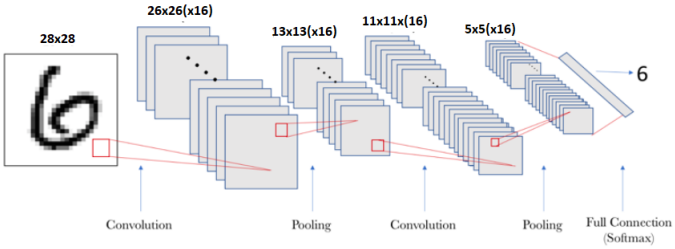

Figure 2: Architecture of the CNN

2.3 Hyper-parameters used for training

Several combination of hyper-parameters was tested, but the best one is shown as follow. In the compile function, it was used the following parameters:

- **loss function**: categorical cross-entropy
- **optimizer**: adam with learning rate equal to 0.001 (default setting)

In order to fitting the model, it was used the following parameters:

- **number of epochs**: 15
- **batch size**: 128

In addiction, it was set the parameter **validation split** equal to 0.1, in order to verify the possible over(under)-fitting of the CNN.

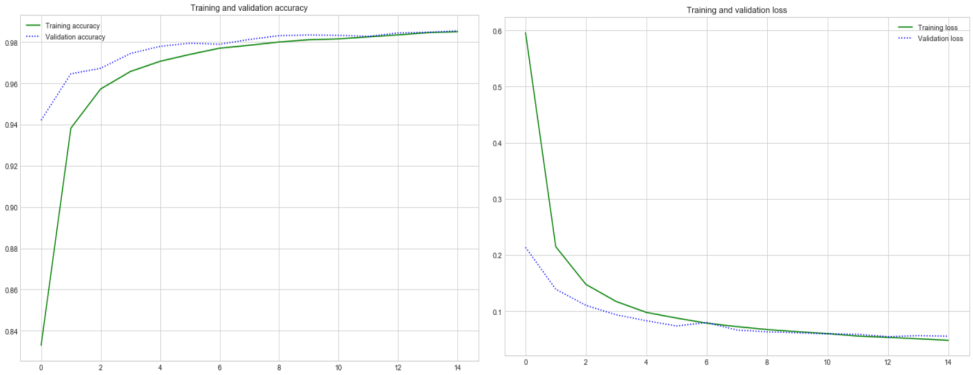

Figure 3: Accuracy and Loss

3 Results

Finally, it was plotted the accuracy and loss function obtained on the training and validation data. From the figure 3 it was possible to assert that the CNN above described worked fine and there were no problem of over (or under) fitting because the values of the accuracy and the loss function converged on the same values.
In conclusion, it was printed the classification report offered by sklearn, where it was possible to observe the values of accuracy on the train and the test. On the training set it was obtained 99% of accuracy and 98% on the test set. The results are shown on the following tables.

Classification Report: TRAIN

	precision	recall	f1-score	support
0	0.99	0.99	0.99	5923
1	0.99	0.99	0.99	6742
2	0.99	0.98	0.98	5958
3	1.00	0.97	0.98	6131
4	0.99	0.99	0.99	5842
5	0.99	0.97	0.98	5421
6	0.99	0.99	0.99	5918
7	0.98	0.99	0.99	6265
8	0.96	0.99	0.97	5851
9	0.98	0.98	0.98	5949
accuracy			0.99	60000
macro avg	0.99	0.99	0.99	60000
weighted avg	0.99	0.99	0.99	60000

Classification Report: TEST

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

Figure 4: Classification report