UNIVERSITY OF PADUA, DEPARTMENT OF INFORMATION ENGINEERING

- Neural Network and Deep Learning -

Handwritten digits classification

Giulia Bressan, ID 1206752

A.Y. 2019/2020

1 INTRODUCTION

The assignment of the second homework is to classify the handwirtten digits provided in the MNIST dataset. The goal is to find the right structure and hyperparameters of a feed-forward neural network, in order to be able to classify the images with high accuracy.
The work is structured as follows:

1. Loading of the given train and test data,
2. Tuning of the parameters of the neural network using k-fold cross validation with grid search,
3. Training of the neural network in the whole set of data,
4. Computation of accuracy metrics and visualization of the receptive field of neurons, results and final considerations.

2 TRAINING AND TEST SET

The data provided for the homework are taken from the MNIST dataset. It is a set of 60 000 images of handwritten digits of shape 28x28 pixels. They were stored in a .mat file as vectors of size 784. In Figure 2.1 an example of some training images is shown, along with the correspondent labels.

Figure 2.1: Examples of training images from the MNIST dataset with correspondent labels.

The images were then divided in training and test set: the 80% of the dataset was used as training set, while the remaining 20% as test set, thus obtaining a 48000 samples for training and 12000 samples for testing the network.

3 Parameters tuning

The chosen network used for the classification task was a feed-forward neural network with one input layer, two hidden layers and an output layer.
In order to properly set the number of neurons for each hidden layer and the learning rate of the network, a k-fold cross validation with grid-search was implemented. The cross validation procedure was implemented only in a subset of the training set consisting of 10000 images, as the training procedure was very time demanding for the whole training set. The subset was then divided in 3 folds and tested for the whole grid of parameters, as shown in Table 3.1, with a fixed number of epochs equal to 10.

Parameter	**Possible values**
First hidden layer	32 - 64 - 128 neurons
Second hidden layer	64 - 128 - 256 neurons
Learning rate	0.01 - 0.02

Table 3.1: Set of hyperparameters used during the cross-validation with grid search.

The results showed that the set which performed better was made of 64 neurons for the first hidden layer and 256 for the second, with a learning rate of 0.01. The input layer had a size of 784, equal to the length of each input vector, and the output layer returned a vector of size 10, equal to the number of possible classes for the digits.
For the training of the network the loss function used was a Cross-Entropy loss function, already defined in the PyTorch framework, and the optimizer was the Adam optimazer, with the chosen learning rate and a weight decay of 0.0005, pre-defined as well. Table 3.2 sums up the optimal set of parameters used for the actual training of the network.

Parameter	**Optimal value**
First hidden layer	64 neurons
Second hidden layer	256 neurons
Learning rate	0.01

Table 3.2: Optimal set of hyperparameters used during the training of the neural network with the entire dataset.

4 Training and testing

The training of the neural network with the whole training set was done with the best set of parameters found with the cross-validation, for 150 epochs. The other parameters, such as loss function and optimizer, were the same as the ones used in the cross-validation.

5 Results

In Figure 5.1 the behaviour of training and test loss during the 150 epochs is shown.
The accuracy obtained with the test set was of the 97%, with 362 errors out of 12000 test samples.

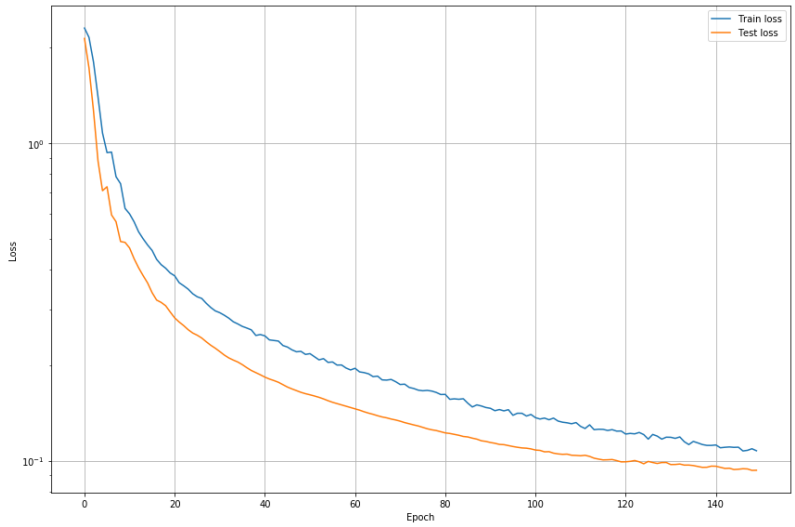

Figure 5.1: Behaviour of training and test loss during the 100 epochs training.

The following confusion matrix shows the distribution of the classification errors in the test set. In the principal diagonal we can see that the majority of the samples are correctly classified, while outside the diagonal we can see the wrongly classified samples. In particular it is interesting to see that 2 and 5 are often classified as 3, while 9 is misread as 4, probably because of the similar shape of the handwritten digits.

$$\begin{bmatrix}
1178 & 0 & 3 & 1 & 2 & 5 & 3 & 0 & 7 & 0 \\
0 & 1263 & 7 & 4 & 3 & 1 & 0 & 2 & 2 & 1 \\
3 & 6 & 1201 & 3 & 5 & 1 & 6 & 8 & 8 & 0 \\
0 & 1 & 19 & 1169 & 0 & 25 & 0 & 10 & 8 & 6 \\
1 & 0 & 2 & 0 & 1135 & 0 & 6 & 5 & 0 & 22 \\
6 & 1 & 1 & 6 & 0 & 1062 & 6 & 0 & 5 & 6 \\
6 & 1 & 1 & 0 & 3 & 5 & 1113 & 0 & 5 & 0 \\
3 & 4 & 8 & 4 & 5 & 1 & 1 & 1279 & 3 & 4 \\
4 & 8 & 3 & 10 & 1 & 7 & 2 & 5 & 1146 & 5 \\
0 & 1 & 0 & 8 & 16 & 6 & 1 & 8 & 6 & 1092
\end{bmatrix}$$

In Figure 5.2 a subset of the wrongly classified digits is shown and we can see that the shapes are not well defined, so the classification may result a bit difficult also for humans.

Figure 5.2: Example of wrongly classified images.

An other interesting feature to show is the receptive field of the neurons for the hidden layers. It is useful to visualize the features learned by the network in each layer.
The receptive fields are computed by combining the weight matrices of the lower layer, thus obtaining and image for each neuron of the layer that shows what part of the original image is detected as a feature by that neuron.
In the following figures (Figure 5.3, 5.4, 5.5) the receptive fields of some random neurons are shown for the first, the second and the output layer, respectively. We can see that the features are not well recognisable, as we do not see clearly, for example, edges or more high level features in the higher layers, and this is due to the implamentation of the network (without L1 sparsity for example).

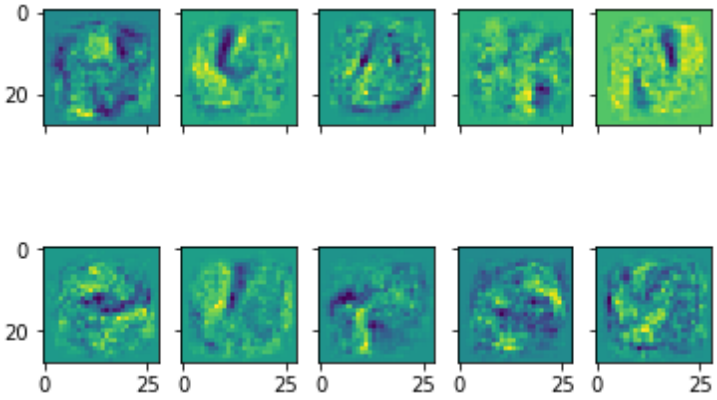

Figure 5.3: Receptive field of a subset of neurons of the first hidden layer.

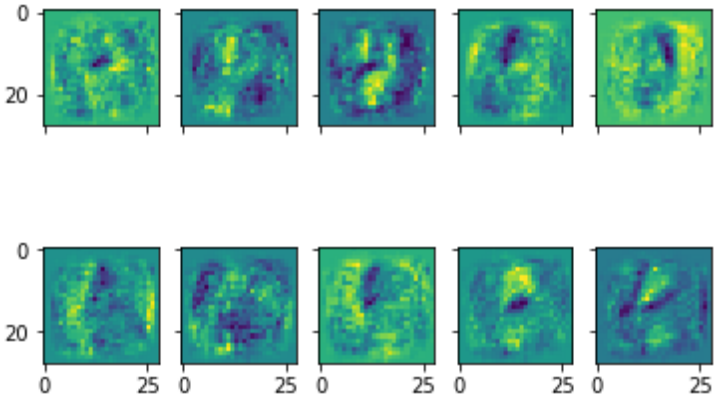

Figure 5.4: Receptive field of a subset of neurons of the second hidden layer.

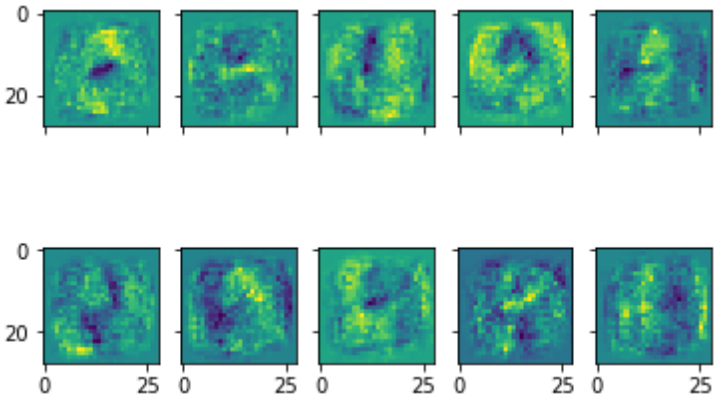

Figure 5.5: Receptive field of a subset of neurons of the output layer.