Assignment: CNN and MNIST

David Bertoldi – 735213
email: d.bertoldi@campus.unimib.it

Department of Informatics, Systems and Communication

University of Milano-Bicocca

✦

1   Inspecting the data

The MNIST dataset contains $70\,000$ images of handwritten digits (0 to 9) that have been size-normalized and centered in a square grid of pixels. Each image is a 28×28 array of floating-point numbers representing grayscale intensities ranging from 0 (black) to 255 (white).

The labels consist of a vector of values, corresponding to the digit classification categories 0 through 9.

The dataset is already divided into training and test sets, respectively with $60\,000$ and $10\,000$ samples.

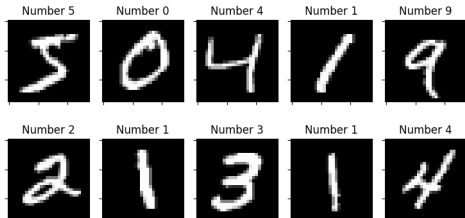Figure 1 shows an example of the population.

Figure 1: The first 10 samples of the train dataset

The training population presents a distribution with mean $\mu = 6\,000$ and standard deviation $\sigma \simeq 340$ and thus we didn't notice any important unbalance in the data. For this reason we assumed the data followed a distribution $X \sim U(\mu, \sigma)$ and no data augmentation on less populated classes was taken into account. Figure 2 shows the data distribution for both training and test datasets.

2   Preparing the data

Before training a FFNN using this images, encoded in 28×28 matrices with values from 0 to 255, we flattened them in arrays 1×784 and rescaled each value in the continuous interval $[0, 1]$. This encoding will be used in every section of this work: a flat array better suits the input layer of a FFNN and small values increases the efficiency in the calculations.

2.1   Data split

As noted in section 1, the dataset is divided into training and test samples. A validation subset is missing and thus
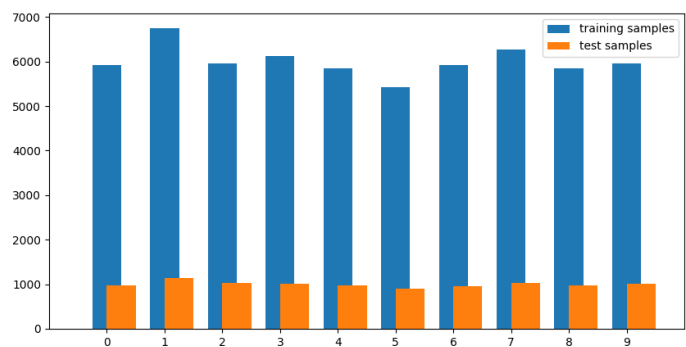

Figure 2: Histogram of the frequency of samples in the dataset

is retrieved from the training set: 15% of the images are randomly used for validation instead of training (along with their labels) for a total of $9\,000$ samples.

About labels, we encoded them in one-hot vectors so that the 1s are set in the index representing the numerical class.

3   Building the network and training

The aim of this section is to describe a CNN with less than $10\,000$ parameters that is able to classify with high level of accuracy the numbers from the dataset with or without any regularization technique.

3.1   The network

The CNN presents a typical architecure formed by convolutional layers followed by pooling layers and ending with dense layers.

In particular there are 2 convulutional layers covering the whole 28×28 matrix, formed by 8 3×3 filters, for a total dimension of $28 \times 28 \times 8$. These two layers are followed by a max pooling layer that halves the the widht and height of the outcoming activation map. For this problem we tested both *max pooling* and *average pooling*; the first one performed slighty better ($+0.1\%$ in test accuracy): usually *average pooling* smooths out the image and the sharp features may be identified with more difficulty, while *max pooling* chooses the white pixels of the image (in case of MNIST dataset, the pixels defining the handwritten digit). Although we noticed a slighty

improvement using *max pooling*, the images are too small to actually benefit from the methods' differences. The structure continues with another one convolutional layer aligned with the 2D spatiality of the last pooling layer but doubled in the depth, that is $7 \times 7 \times 16$. The convolutional layer is reduced in spatiality by another *max pooling layer* $7 \times 7 \times 16$ and flattened in a 1D array of 784. The input flows to an output layer activated by *Softmax* function. Figure 3 summarizes the entire architecture and Table 1 highlights the number of parameters in each layer.

Layer	Size	Parameters
input	$28 \times 28 \times 1$	0
Conv2D-1	$28 \times 28 \times 8$	$(3 \cdot 3 \cdot 1 + 1) \cdot 8 = 80$
Conv2D-2	$28 \times 28 \times 8$	$(3 \cdot 3 \cdot 8 + 1) \cdot 8 = 584$
MaxPool-1	$14 \times 14 \times 8$	0
Conv2D-3	$14 \times 14 \times 16$	$(3 \cdot 3 \cdot 8 + 1) \cdot 16 = 1\,168$
MaxPool-2	$7 \times 7 \times 16$	0
Flatten	$1 \times 1 \times 784$	0
Dense	$1 \times 1 \times 10$	$(784 + 1) \cdot 10 = 7\,850$
Total		$9\,682$

Table 1: Summary of the layers' dimensions and count of the parameters for each layer

The only regularization used is the *dropout* technique applied to the flattened layer with a rate of 40% and *early stopping* during validation. This allowed the model to not overfit too much and generalize better the problem.
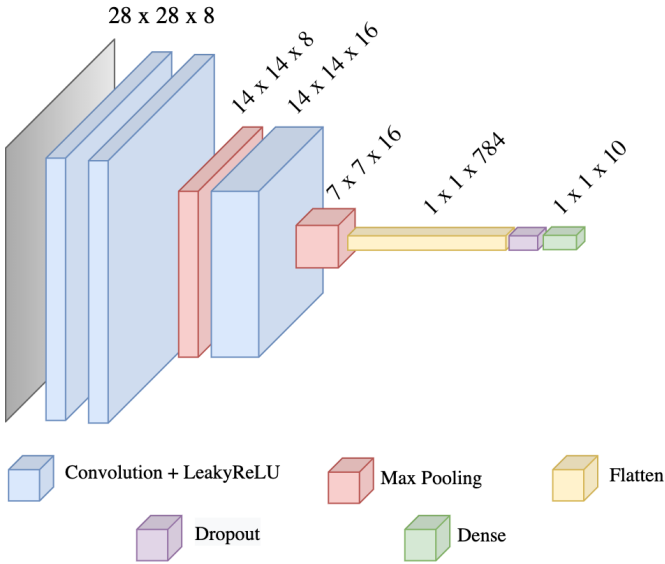

Figure 3: Architecture of the CNN

3.2 Training

The choice of the optimizer was among *SGD* and *Adam*. As expected the first one performed better with small batches (2 to 4) and the second one with batches of 256 samples, but *Adam* performed better overall, with +1% on test accuracy.