2EL1730: MACHINE LEARNING

CENTRALESUPÉLEC

Lab 8: Convolutional Neural Networks for Hand-written Digit Recognition

Instructors: Nora Ouzir and Maria Vakalopoulou
TAs: Theodore Aouad, Hakim Benkirane, Aaron Mamann, Lily Monnier

January 13, 2023

1 Description

The goal of the lab is to implement a general purpose Convolutional Neural Network and to apply it to the task of hand-written digit recognition. Nowadays, automated handwritten digit recognition is widely used – from recognizing zip codes (postal codes) on mail envelopes to recognizing amounts written on bank checks. For the lab needs, we will use the MNIST database[1] which consists of digits written by high school students and employees of the United States Census Bureau. Actually, the MNIST dataset consists of handwritten digit images $(0-9)$ and it is divided in 60.000 examples for the training set and 10.000 examples for testing. Figure 1 represents a randomly selected instance of the dataset.

Figure 1: Example of one hand-written digit of the training set.

All digit images have been size-normalized and centered in a fixed size image of 28×28 pixels. Each pixel of the image is represented by a value in the range of $[0, 255]$, where 0 corresponds to black, 255 to white and anything in between is a different shade of grey. That way, the training set has dimensions $60,000 \times 28 \times 28$ and the test set $10,000 \times 28 \times 28$. Regarding the class labels, each figure (digit) belongs to the category that this digit represents (e.g., digit 2 belongs to category 2).

2 Convolutional Neural Networks - Pipeline

The main goal of the lab is to give us the opportunity to examine and analyze the behavior of different Convolutional Neural Networks structures. In general terms, you should examine and compare the performance of Convolutional Neural Networks with different number of *hidden layers* and *convolutions* per layer, in the problem of hand-written digit recognition.

[1]The MNIST database: `http://yann.lecun.com/exdb/mnist/`.

For notation simplicity, let's assume that we have a relatively small neural network that follows the LeNet architecture with only 2 convolutional and 2 fully connected layers of the following structure:

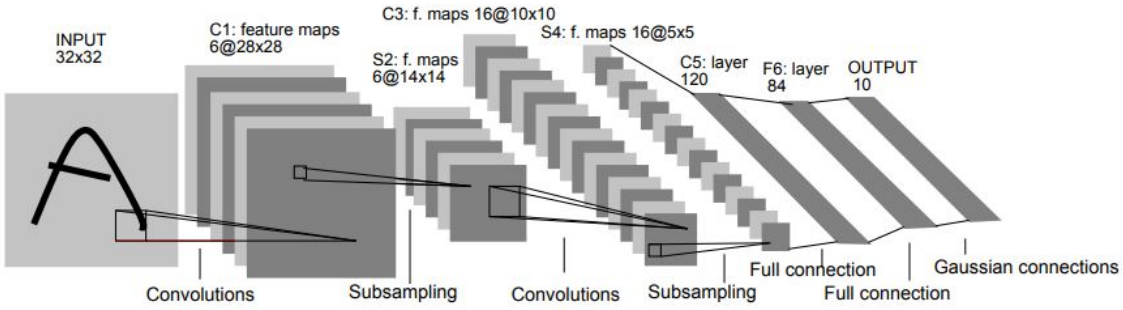

Figure 2: The LeNet Architecture

2.1 Creating the proper environment and install requirements

After you download the exercise please open a terminal in that folder and run the following commands:

```
pip install virtualenv
python -m virtualenv dl-env
source dl-env/bin/activate
pip install -r requirements.txt
python -m ipykernel install --user --name=dl-env
jupyter notebook
```

After opening the notebook make sure to change the kernel to the one named `dl-env` by using the following drop-down menu `Kernel > Change kernel > dl-env`

2.2 Loading and Visualizing the MNIST Dataset

In the first part, the code will load the MNIST dataset, calling the already implemented function in tensorflow `tf.keras.datasets.mnist.load_data()`. This returns the variables `x_train` and `y_train` which contain the training instances and their class labels, respectively. In a similar way, the test data and their class are loaded in the variables `x_test` and `y_test`, respectively. Recall that there are $60,000$ examples for the training set and $10,000$ examples for testing, where each instance is a 28×28 pixels grayscale image of the digit. Each pixel of the image is represented by a floating point number and we normalise them in the range of $[0, 1]$ indicating the grayscale intensity at that location. Finally, we use the function `tf.keras.utils.to_categorical` to create a binary matrix representation of the input. The classes axis is placed last.

2.3 Setting up the Convolutional Neural Network Structure

In the second part, you are asked to set up the structure of the Convolutional Neural Network under which you intend to train for the hand-digit recognition task. Actually, you will be asked to select all the hyperparameters of the network. The cost function of a neural network is calculated according to the following formula:

$$J(\Theta) = \frac{1}{m} \sum_{i=1}^{m} \sum_{k=1}^{K} \left[-y_k^{(i)} \log((h_\theta(x^{(i)}))_k) - (1 - y_k^{(i)}) \log(1 - (h_\theta(x^{(i)}))_k) \right], \tag{1}$$

where $K = 10$ is the number of possible labels and m is the total number of training instances. Note that $h_\theta(x^{(i)})_k$ represents the activation of the k^{th} output unit.

Tasks to be Performed

- Use the tensorflow functions: `tf.keras.layers.Conv2D, tf.keras.layers.MaxPooling2D` and `tf.keras.layers.Dense` to generate the LeNet architecture. [2]

  Fill in the architecture in the Model Definition cell of your jupyter.

- Use the functions `model.compile` and `model.fit` to run the model. Take some time to see the syntax of these functions and the arguments that you need to specify.

- Please indicate the approximate number of epochs that are needed for the architecture to be trained properly.

2.4 Evaluate the training

In the third part of the exercise you will have to print the training curves to evaluate the training. Use the `history` to print the training and validation curves for the accuracy and loss.

Tasks to be Performed

- Use the `model.evaluate` function to run the test dataset and print its loss and accuracy.

- Use the `model.predict` function to extract the predicted classes for the test dataset. Calculate the confusion matrix. What do you observe? Which classes had the higher errors?

2.5 Visualise the kernels of the convolutional network

In the forth part of the exercise you will have to visualise the kernels of the network that have been trained.

Tasks to be Performed

- Visualise the kernels of the first layer. You can use the `get.weights` function of tensorflow.

- Perform the same for the second layer.

2.6 Modify the architecture

In the forth part of the exercise you will be free to perform any modification to the network you think it is appropriate and compare the differences in the performances.

2.6.1 Activation Functions

The original paper of LeNet uses `Tanh` as an activation function.

Tasks to be Performed

- What other activation function could be used?

- Perform a training using `ReLU` activation function. How this affected the training of the model?

2.6.2 Dropout layers

Modify the architecture by adding dropout layers.

Tasks to be Performed

- Use the same architecture and add dropout layers. Use the `Dropout` function of keras with different dropout rates.

[2] The LeNet paper: `http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf`.