

Neural networks LAB3

Deep Neural Networks

The goal of this LAB is to understand the architecture of two major neural networks :

- Standard deep neural networks
- Convolutional neural networks

Part 1 – Dataset

In this lab, we will solve a problem where samples have many more features : images. This convoluted problem will require deeper, more complex neural networks to solve.

1. Load the MNIST dataset. How many training and test images are there? How many classes are there? How many features do we have? Display the first image of the training set.

2. Display the number of training samples in each class.

In the MNIST dataset, the images are a tensor in the shape of [60000, 28, 28]. The first dimension is used to extract images, and the second and third dimensions are used to extract pixels in each image. Each element in this tensor indicates the strength of a pixel in an image. The value ranges from 0 to 255.

3. For the two parts to come, preprocess the data by flattening out the image. This means that each image becomes an n^2 -length vector, instead of an $n \times n$ matrix. Also, normalize the image pixels so that they range from 0 to 1 instead of 0 to 255.

4. The reason we flatten out the image is that fully-connected neural networks cannot use 2D input, but that doesn't mean it is standard behaviour. Explain why, in reality, flattening out images before classifying them sounds like a bad idea.

Part 2 – Linear model

5. Implement ten neural networks with no hidden layers, as was done in the previous lab (use SGDClassifier). Train them on the training data, test them on the test set, and give the mean and standard deviation of the test accuracy. Why is it useful to do this before doing the following parts?

Part 3 – A deep neural network

From this part until the end of the course, we will switch to another library : Keras. Keras allows us to implement deep neural networks easily.

6. Use the Keras cheat sheet to implement a neural network consisting of a succession of :

- an Input layer with a shape of however many input features there are
- a Dense layer with 512 neurons
- a ReLu activation layer
- a Dense layer with 256 neurons
- a ReLu activation layer
- a Dense layer with 128 neurons
- a ReLu activation layer
- a Dense layer with 64 neurons
- a ReLu activation layer
- a Dense layer with 32 neurons
- a ReLu activation layer
- a Dense layer with 16 neurons
- a ReLu activation layer
- a Dense layer with 10 neurons
- a Softmax activation layer

Compile the model with the sparse categorical crossentropy loss, the Adam optimiser with learning rate 0.001, and 'accuracy' as the only metric.

7. Why do we use the Softmax activation layer at the end?

8. Fit the neural network on the training data (use epochs of 10 and a batch size of 60). Print the test accuracy.

9. Get the summary of the total amount of parameters in the DNN (use its **summary()** method), and explain as much of the table as possible.

10. Save the model on your computer.

Part 4 – A Convolutional Neural Network

11. Use the Keras cheat sheet to implement a Convolutional neural network with :

- a 2D input (not the one you used in the previous section !)
- a 2D Convolution layer with 32 filters, a kernel size of 5, strides of 1 for each dimension, and the padding set to 'same'
- a ReLu activation layer and a max pooling layer with 2x2 size, 2x2 strides and padding set to 'valid'
- a 2D convolution layer with 64 filters, a kernel size of 3, and the same strides and padding as before

- the same ReLu and MaxPool layers as before
- a flattening layer
- a 128-neuron Dense layer
- a Relu layer
- a 10-neuron Dense layer
- a Softmax layer

Compile it using the same parameters as previously.

12. Fit the model with a batch size of 128, and 5 epochs. What is the test accuracy this time?

13. Get the total amount of parameters in the CNN and do your best to explain as much of it as possible.

14. Display the first 5 images of the test set along with and print the predicted class for each of them.

15. Use the model to do new predictions! Load the four images, resize them to the shape neural networks can accept, and make sure the images are normalized correctly. Then, predict their label, and show them next to their label.