Introduction to deep learning

Timon Deschamps, original document by Mathieu Lefort

September 9, 2025

This project can be done in pair working (in this case you will precise in your report what each one of you did). You have to provide a report including your code and your answers to the questions at the end of each session and the final version before October 9. Do not forget to put your(s) name(s) in the uploaded file.

1 Aim

The aim of the project is to implement the MLP and CNN models seen during the lectures, to understand how they work and to become familiar with the PyTorch framework.

2 Installation

You have to install python, pytorch and numpy (and matplotlib if you intend to do some graphs/plots). To install PyTorch, have a look here :`https://pytorch.org/get-started/locally/`. To check if installation was successful, open a python terminal and type the following command : `import torch`.

3 PyTorch

You will find the list of available functions here : `https://pytorch.org/docs/stable/index.html`. Have a look to this mini tutorial : `http://pytorch.org/tutorials/beginner/pytorch_with_examples.html`. Key concepts are presented here : `https://pytorch.org/tutorials/beginner/basics/intro.html` (you can skip chapters 3 and 7).

4 Data

You have at your disposal the MNIST dataset than contains images (with size of 28×28) of hand written digits (e.g. here is an image of a 5 in the dataset : ). The dataset has the following structure : ((train_images_array, train_labels_array), (test_images_array, test_labels_array)). The images are represented as a vector of dimension 784 (=28*28), and the labels are in one-hot encoding (e.g. if the image corresponds to a 5, the label will be : $[0, 0, 0, 0, 0, 1, 0, 0, 0, 0]$).

Thus, the input layer of each network has 784 dimensions (28×28 pixels) and the output layer has 10 dimensions (the i^{th} value represents how much the network

recognizes the digit i in the image). This is a classification problem, therefore you will have to use an appropriate loss function.

Alternatively, you can use the fasion MNIST dataset, which shares the same structure, image size, and number of labels as the original MNIST, but contains images of clothing articles for an added challenge.

5 Part 1 : Perceptron

You have at your disposal two files implementing (the same) perceptron (presented during the lessons) :

1. PERCEPTRON_PYTORCH.PY which uses only tensors
2. PERCEPTRON_PYTORCH_DATA_AUTO_LAYER_OPTIM.PY which uses most of PyTorch tools (dataloaders, automatic gradient, layers and optimizers)

— Indicate and explain the size of each tensor of the provided file PERCEPTRON_PYTORCH.PY.

6 Part 2 : Shallow network

In this part, you will implement a MLP with only one hidden layer and a linear output layer.

— Implement a shallow network using the tools provided by PyTorch.
— Describe very precisely the methodology to use (do not forget to create a validation dataset to avoid over fitting, and remember that initial weights are random) to find the hyperparameters (η, number of hidden neurons, batch size) that give the best performance.
— Find hyperparameters that provide a reasonably good performance (you have to adapt the number of runs from your methodology to your actual computational power, and explain/justify why you choose to perform these tests). Explain the influence of each hyperparameter on the performance.

7 Part 3 : Deep network

As you are now more familar with PyTorch, you can use it to test deeper models.

— Implement a deep network (i.e. with at least two hidden layers) using the tools provided by PyTorch.
— Find hyperparameters (η, number of hidden layers, number of neurons in the hidden layers and batch size) that provide a reasonably good performance (you have to adapt the number of runs from your methodology to your actual computational power, and explain/justify why you choose to perform these tests). Explain the influence of each hyperparameter on the performance.

8 Part 4 : *CNN*

As you have now a deeper understanding of the MLP, it's time to implement a CNN architecture that is more adapted to images (pay attention that data are with a format of a vector with dimension 784 and that you have to transform them in images of size 28×28).

— Implement a CNN using the tools provided by PyTorch that gives you good performances. You can base your model on a simple architecture such as LeNet5 or fine tune a pretrained model such as ResNet. Explain and justify your approach (w.r.t. your available computational power).