Created by Florian Wolf; Questions, errors and feedback via GitHub (preferred) or email.

Deep Learning Workshop, 27.03 – 30.03.2023

Assignment 00: General Concept and Data Exploration (Preparation)

General Concept of the workshop

Every Machine Learning Workflow can be broken down into (more or less) four main tasks:

1) Prepare the dataset.

2) Choose a model type and a specific architecture for that model class.

3) Train the model on the training data and evaluate it on the validation data.

4) Evaluate the model's performance on (unseen) testing data and use the model as a predictor.

In every assignment we will work on these four components that are also visualized in Figure 1. To implement a good coding style and simplify the process for all the assignments in general, we will encapsule the four different components by separating model specific and model independent parts. Every assignment will use and consist of the four following components:

Data Loader [DL] (Model dependent) In each assignment we will either implement a custom Data Loader class by ourselves or use an already implement PyTorch version. In general, the Data Loader provides the training, validation and testing dataset. The Data Loader is model dependent, as e.g. the size of the data samples has to match the size of the network and we usually perform the model dependent data augmentation inside the Data Loader class. Thus, we will always pair a network with its corresponding Data Loader.

Network [N] (Model dependent) In order to solve a Machine Learning task, we need to decide on a specific model type (e.g. Fully Connected Neural Network) and on the network's architecture (e.g. number of hidden layers). As mentioned above, the Network and the Data Loader always determine each other.

Network Trainer [NT] (Model independent) The Network Trainer class takes a Network, a Data Loader, a loss function and an optimization algorithm as parameters and trains the model independently from it's structure. Furthermore, the Network Trainer automatically saves the parameters of a network so we can later reload the network and thus decouple the training and evaluation process.

Network User [NU] (Model independent) The Network User class is the last component of our generic Machine Learning pipeline. First, it loads the trained model, evaluates the network on the testing dataset and creates plots of the model's performance. Additionally, we have the option to obtain a function acting as a predictor, to use the model for predictions on unseen data points. Similar to the Network Trainer, the Network User class works independently of the model's structure.

Although this approach has the disadvantage that in the beginning, we have to deal with a higher level of abstraction and thus the code is more difficult to read, we also obtain a lot of advantages by using this programming style:

- We produce less errors and our code is easier to debug, as we decouple different parts of the pipeline.
- If the training process of a networks works, but the evaluation fails, we do not have to train the network again, but instead can just solve the issue in our evaluation before running the evaluation again. This saves a lot of time and nerves.
- We can easily modify our code to scale the architecture to larger systems.
- The resulting code framework can be afterwards used by you as a starting point for your personal Deep Learning projects.

In the assignments, we will use the aforementioned abbreviation for the different parts but *don't worry*. We will revise the framework in the beginning of the workshop and each subtask will be annotated with the structure we will use or implement. Now you are ready to start working on the data exploration tasks and make yourself familiar with the data for the first assignment. Again, the main goal of this first exercise is to make yourself familiar with the dataset and the general structure we will use during the workshop.

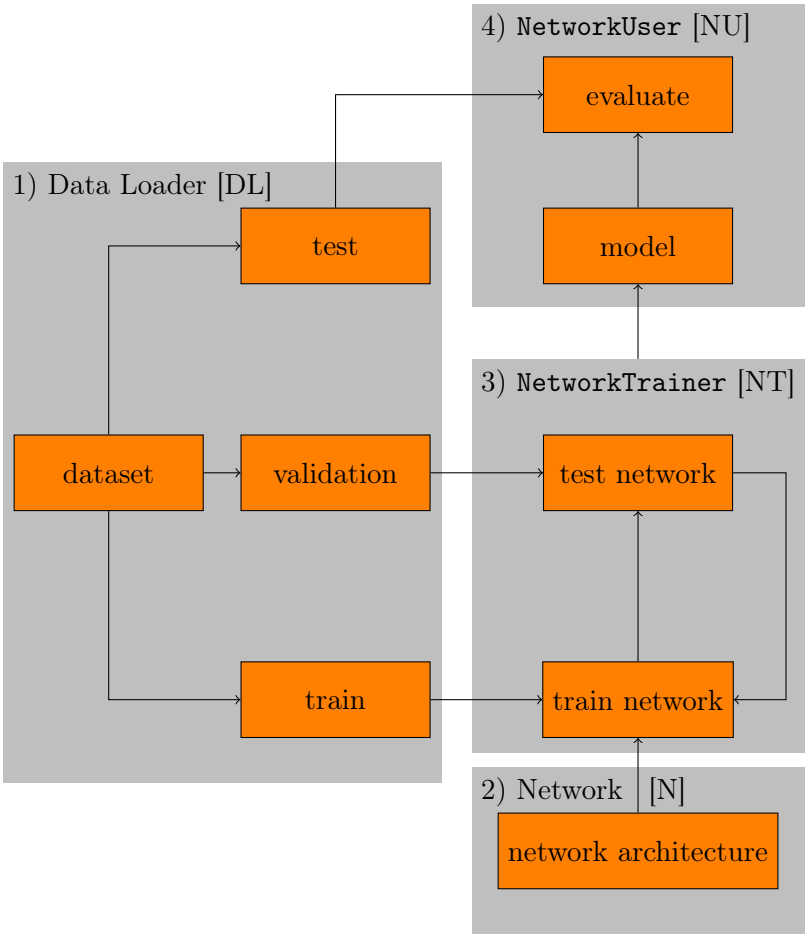

Figure 1: Generic Machine Learning Pipeline including the different components we will work on during the workshop.

Preparation Task

1) **Data exploration: Visualizing what the machine will see.**
In this task we will work with the files `00_main.py` and `fnn.py`.

   a) [DL] Open the file `fnn.py` and you should find two class definitions: `FNN` consisting of a blank template (we will work on that in the workshop's first assignment) and `DataLoaderFNN`. The latter will provide the training, validation and testing dataset. The internal structure of the `DataLoaderFNN` class is at this point not important, but you can check the implementation if you are interested. You obtain the training, validation and testing datasets by instantiating the class and calling it via

```
data_loader = DataLoaderFNN(train_val_ratio=0.8, batch_size=100)
train_data = data_loader("train") # "val" and "test" respectively
```

   see also in the `00_main.py` file. Remove the `pass` statements and implement a simple method in your `00_main.py` file to load 6 images from the train dataset, their corresponding labels. Visualize the images in a plot by using the `show_images` method. Analyze the shape of the given data, what do you observe?

   b) Iterate through all existing labels in the training, validation and testing dataset and visualize the frequency each label has (e.g. by using a histogram or bar chart).

Now we are familiar with the data, our code works locally and we are ready to implement our first network in the workshop. We look forward to working together with you in the workshop, see you soon!

References

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.