# *Machine Learning A*

*2023-2024*

## Home Assignment 6

**Christian Igel**
Department of Computer Science
University of Copenhagen

The deadline for this assignment is **12 October 2023, 22:00**. You must submit your *individual* solution electronically via the Absalon home page.

A solution consists of:

- A PDF file with detailed answers to the questions, which may include graphs and tables if needed. Do *not* include your full source code in the PDF file, only selected lines if you are asked to do so.
- A .zip file with all your solution source code with comments about the major steps involved in each question (see below). Source code must be submitted in the original file format, not as PDF. The programming language of the course is Python.
- IMPORTANT: Do NOT zip the PDF file, since zipped files cannot be opened in *SpeedGrader*. Zipped PDF submissions will not be graded.
- Your PDF report should be self-sufficient. I.e., it should be possible to grade it without opening the .zip file. We do not guarantee opening the .zip file when grading.
- Your code should be structured such that there is one main file (or one main file per question) that we can run to reproduce all the results presented in your report. This main file can, if you like, call other files with functions, classes, etc.
- Handwritten solutions will not be accepted. Please use the provided latex template to write your report.

# 1 Logistic regression in PyTorch (25 points)

You are supposed to fill in the missing lines in the notebook `Logistic Regression PyTorch Assignment.ipynb`. The tutorial `https://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html` should provide you with all knowledge you need.

The notebook considers an example where 2D data points are classified using logistic regression and the solution is visualized. First, an implementation in Scikit-Learn is given. Then the same is partly implemented using PyTorch. Your assignment is to fill in the blanks in the PyTorch implementation.

You should add some code in several cells (indicated by "MISSING"). Please write these lines of code in your report, see the LaTeX template.

# 2 Convolutional Neural Networks

## 2.1 Sobel filter (25 points)

In this assignment, you should implement a Sobel filter in PyTorch. The learning goals are a better understanding of basic image filtering operations, a deep understanding of how the PyTorch convolutional layers work, and better PyTorch programming skills.

The Sobel filter is a basic operator to highlight edges in images (e.g., see Wikipedia). It convolves the image with two $3 \times 3$ filters, the results of which are combined. Let the matrix $\boldsymbol{I}$ represent a (single-channel) 2D input image and $*$ the convolution operator. Then the Sobel filter computes in a first step

$$\boldsymbol{G}_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} * \boldsymbol{I} \qquad \text{and} \qquad \boldsymbol{G}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \boldsymbol{I} \ , \tag{1}$$

which are then combined to a feature map

$$[\boldsymbol{G}]_{ij} = \sqrt{[\boldsymbol{G}_x]_{ij}^2 + [\boldsymbol{G}_y]_{ij}^2} \ , \tag{2}$$

where $[\cdot]_{ij}$ denotes the matrix element at position $ij$.

Start by going through the notebook `PyTorch 2D convolutions.ipynb`, which gives some examples on how to work with 2D convolutions in PyTorch and provides some basics on how to read and display images. Do this carefully. First, it is surprising how many people doing deep learning in Python actually do not know what is going in a simple convolutional layer. Second, handling input images in


Figure 1: Examples from the traffic sign data set.

real-world applications can sometimes be tricky (and I am not claiming that the way I did it in the notebook is the most elegant one).

Implement the Sobel filter. Use a single `nn.Conv2d` call to generate the two feature maps $\boldsymbol{G}_x$ and $\boldsymbol{G}_y$. For squaring each element in a filter output you can use something like `torch.square`. For combining filter outputs as in (2) you can use `torch.sum`, but you have to set the `axis` argument properly.

*Deliverables:* Show the important parts of your code (i.e., the Sobel filter) in the report. Apply the Sobel filter to the same input image as in `PyTorch 2D convolutions.ipynb`. *Deliverables:* Show plots of the two feature maps and the final result (i.e., $\boldsymbol{G}_x$, $\boldsymbol{G}_y$, and $\boldsymbol{G}$).

*Deliverables:* In the report, discuss how you considered the difference between convolution and cross-correlation (note that the latter is what the "convolutional" layer is doing) in your implementation.

## 2.2 Convolutional neural networks (25 points)

The learning goal of this part of the assignment is to get more comfortable with implementing convolutional neural networks (CNNs) in PyTorch.

Consider the notebook `Torch_Traffic_Signs_Basic_Template.ipynb` and the file `GTSRBTrafficSigns.py`. The notebook is a template for training a convolutional neural network (CNN) for classifying traffic signs on the same data as the competition described by Stallkamp et al. (2012). The data is handled by the `Dataset` class in the file `GTSRBTrafficSigns.py`. Recognition of traffic signs is a challenging real-world problem of high industrial relevance. Traffic sign recognition can be viewed as a multi-class classification problem with unbalanced class frequencies, in which one has to cope with large variations in visual appearances due to illumination changes, partial occlusions, rotations, weather conditions, etc. However, humans are capable of recognizing the large variety of existing road signs with close to 100 % correctness – not only in real-world driving situations, which provides both context and multiple views of a single traffic sign, but also when looking at single images as those shown in Figure 1. Now the question is how good a computer can become at solving this problem.

The notebook is supposed to run with GPU support, for example, using *Google Colaboratory*. Even then, executing it will take some time.

The notebook misses the CNN definition. Define a network with a

- convolutional layer creating 64 feature maps using $5 \times 5$ kernels followed by a
- $2 \times 2$ max-pooling layer followed by a
- convolutional layer creating 64 feature maps using $5 \times 5$ kernels followed by a
- $2 \times 2$ max-pooling layer followed by a
- linear fully-connected layer with 43 outputs.

After each convolutional layer, the ELU (exponential linear unit, Clevert et al. (2016)) activation function should be applied.

*Deliverables:* Show the model definition in the report.

You should get the following output when you print your model:

```
Net(
  (conv1): Conv2d(3, 64, kernel_size=(5, 5), stride=(1, 1))
  (pool1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
   =1, ceil_mode=False)
  (conv2): Conv2d(64, 64, kernel_size=(5, 5), stride=(1, 1))
  (pool2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation
   =1, ceil_mode=False)
  (fc2): Linear(in_features=1024, out_features=43, bias=True)
)
```

## 2.3 Augmentation (25 points)

Convolutional neural networks are highly complex models. To reduce the risk of overfitting and to be able to learn difficult tasks with a high input variability, many training examples are needed. *Data augmentation* is a way to enlarge the training data set by artificial training points generated from the available data. Data augmentation refers to applying transformations to the input data (the images) that do not change their labels. If we know, for example, that the classifications of images do not change when the images are rotated, then showing rotated versions of the images during the training process helps to learn this invariance property. In the context of neural networks, this type of data augmentation can be traced back to Baird (1992). It has been used successfully for CNNs, for instance, already in the influential work by Krizhevsky et al. (2012)

The learning goal of this part of the assignment is to get more experience in using data augmentation.

Inspect the notebook `Torch_Traffic_Signs_Basic_Template.ipynb`.

*Deliverables:* Answer the following questions briefly in your report: Which transformations are applied to the input images during training? Why is a transformation conditioned on the label?

Please add at least one additional (not completely nonsensical) transformation.

*Deliverables:* Show the corresponding code in your report and briefly argue why you think that this may be a reasonable augmentation given the task.

# References


Henry S. Baird. Document image defect models. In *Structured Document Image Analysis*, pages 546–556. Springer, 1992.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (ELUs). In Yoshua Bengio and Yann LeCun, editors, *International Conference on Learning Representations (ICLR)*, 2016.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2012.

Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32:323–332, 2012.