1 Practical 1

1.1 Information and instructions

1. DEADLINE: Wednesday 24 September 2025 before 23:59. Hand in your deliverables containing your code and report through Brightspace using the provided LaTeX template.
2. This is a group assignment to be completed in pairs.
3. Use formulas when necessary to answer a question as well as a short description of how you come up with your solution.
4. Please ensure that all your figures are properly labeled with a title, axis labels, and a legend if you plot multiple curves in one figure.
5. Structure your code using comments and separate code cells where necessary, and make sure to indicate which part of your code belongs to which question.
6. To test your code, we will run all your code from scratch - so make sure all results can be reproduced! For any questions, you can contact us at j.guo@rug.nl.

1.2 Exercise 1 - Get a running VGGNet for image classification on CIFAR-10 dataset [3pt]

In this exercise, you will get hands-on experience in Convolutional Neural Networks (CNNs), e.g., the VGGNet, using PyTorch.

On Brightspace, you will find the file `VGG_cifar10_lab1.zip`, which contains the PyTorch implementation of VGGNet. The code is partly taken from a Github repository [1]. If you are interested in the full version with more examples of other CNN architectures, please check the repository.

You are highly recommended to use Hábrók to set up your experimental environment using dedicated GPU nodes from computer science. Here is an example on the shell script to schedule your gpu job in Hábrók with name `jobscript.sh`. This file is also included in the zip file. For the explanation of the parameters, please check the instructions on brightspace or on Hábrók wiki.

```
#!/bin/bash
#SBATCH --partition=digitallab
#SBATCH --nodes=1
#SBATCH --gpus-per-node=1
#SBATCH --mem=2GB
#SBATCH --time=01:00:00

# start environments and load existing modules
module load Python/3.10.4-GCCcore-11.3.0
module load PyTorch-bundle/2.1.2-foss-2023a-CUDA-12.1.1

# run your Python main code here
python3 main.py
```

[1] https://github.com/kuangliu/pytorch-cifar

In order to submit the job, you need the command `sbatch jobscript.sh`. You can view, monitor, or cancel your jobs with `squeue -u <your s-number>`, `jobinfo <JobID>`, or `scancel <JobID>`.

Plot the results you get in terms of loss and accuracy for both the training and test sets as the number of epochs (e.g. `50` epochs) increases (one figure for the loss and one for the accuracy). The `x`-axis denotes the epoch number while the `y`-axis indicates the corresponding loss and accuracy.

1.3 Exercise 2 - Understand and tweak the code [3pt]

1. Data-related questions:
a. Briefly describe the CIFAR-10 dataset used in this implementation, incl. the types of images and categories.
b. What are the transformations used for data augmentation?

2. Model-related questions:
a. What is an optimizer? What is the initial learning rate? You can change the value of the initial learning rate and check how it affects the model performance (fast or slow convergence). You can select two different learning rates, one is larger and the other is smaller than the default setting and plot the training curves.
b. Confirm the size of the feature maps after each convolution block. Display the feature maps. You can show 3 images of the 2D feature maps from 3 different convolutional blocks: the first, the last and one in between that is up to your choice. In total, you will have 9 images.

1.4 Exercise 3 - Construct your own CNN for image classification [4pt]

Replace the VGGNet with a convolutional neural network designed by yourself. This network should have **four** layers with trainable parameters. You have the complete freedom to design the network, e.g. using different activation functions (tanh, leakyReLu, etc.), trying dropout layers (with `torch.nn.Dropout`), etc..

1. Make a figure to describe the architecture of your CNN. Specify the corresponding hyper-parameters and trainable parameters used in every layer of your network. Calculate manually the output size of each layer.

2. What is the total number of trainable parameters of your network? Report the results you get after certain number of epochs (e.g. `50`) and compare them with the results obtained from VGGNet.