3 Practical 3

3.1 Information and instructions

1. DEADLINE: Wednesday 5 February 2025 before 23:59PM. Hand in your report using the provided LaTeX template and your source codes (.py files) through Brightspace in one zip file.
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.
6. 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.
7. Structure your code using comments and separate code cells where necessary, and be sure to indicate which part of your code belongs to which question.
8. 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 patternrecognition.cs.rug@gmail.com.

3.2 Exercise 1 - Generative adversarial network (70pt)

In this assignment, you will implement a Generative adversarial network (GAN) for natural image synthesis. As introduced in the lecture, GANs consist of two networks, namely a generator and a discriminator, which compete with each other to generate synthetic instances of data that mimics real data. In `GAN.py`, the general framework of a GAN is provided, and you will need to implement the missing parts in the code.

Step1 (40pt) - The GAN generator learns to create fake data from random noises. It takes as input the noise vector of `nz`-dimensions and outputs an image of size 32×32. The generator architecture consists of several transposed convolutional blocks or convolutional blocks with upsampling operations. Please provide the missing codes for the generator with **four** trainable layers. Make sure the final output has the appropriate range of pixel values (e.g., $[-1, 1]$) by selecting a proper activation function after the last trainable layer. Briefly explain what are the outputs after every operation.

Step2 (20pt) - The GAN discriminator learns to distinguish between the generated fake images and the real images. It takes as input an (fake or real) image and output its label (0 or 1). The architecture of the discriminator consists of multiple strided convolutional blocks or convolutional blocks with maxpooling operations. In the following, please provide the missing codes for the discriminator with **four** trainable layers. You can use the 4-layer network designed by yourself in Lab 2 but you need to adjust the output of the last layer. Again pay attention to the output values of the discriminator in order to use the proper activation function after the last trainable layer. Briefly explain what are the expected outputs after every layer.

Step 3 (10pt) - Train the GAN using the completed generator and discriminator. Plot the loss curves for the generator and discriminator over the training epochs. Additionally, visualize the generated images at various stages of training to assess the progress

of the generator. Include these plots and images in your final report.

3.3 Exercise 2 - Conditional GAN (30pt)

In this exercise, you will extend the GAN framework to implement a conditional GAN (cGAN) in a new file `cGAN.py`. cGANs enable the generation of category-specific images by conditioning both the generator and discriminator on additional information, such as class labels. In this task, the class labels are encoded as one-hot vectors.

For the generator, the associated class labels (one-hot encoded vectors) are concatenated with the noise signal. The input vector to the generator will have a length equal to the sum of the noise vector length and the length of the one-hot encoded class labels. Modify the generator in `GAN.py` accordingly.

For the discriminator, the associated class labels are concatenated with image embeddings (vectors), and then input together to the last fully connected layer for final classification. Update the discriminator in `GAN.py` to incorporate this change.