HY673 - Spring 2024 Problem Set #4
Yannis Pantazis (13/05/2024)

Instructions

- **Due date:** Monday June 10th, 2024
- Submission via e-mail to the class account: hy673@csd.uoc.gr
- Provide one file with the written solutions.
- Provide one folder with code.
  - The name of each file in the folder should indicate the respective exercise.
  - Your code should run on colab.
- All assignments in this course are individual, not group, assignments. You may freely discuss homework assignments with your fellow classmates. The final solutions, however, must be written entirely on your own. This includes programming assignments.
- You are allowed to use Generative Al Tools such as ChatGPT for help with homework assignments for grammatical corrections only. To maintain academic integrity, students must disclose any use of Al-generated material.

Problem 1 (Train GANs on the Fashion MNIST dataset)**.** *In this exercise you will generate images from the Fashion MNIST dataset. The FashionMNIST can be simply imported using the torchvision module via:*

```
from torchvision.datasets import FashionMNIST
dataset = FashionMNIST(rootdir, train=True, download=True)
```

(a) Train a deep convolutional generative adversarial network (DCGAN) using the original loss function on Fashion MNIST dataset. The architecture of the generator will consist of one dense and three convolutional layers while the architecture of the discriminator will consist of three convolutional and one dense layer (i.e., reverse order)[1]*. Use Adam optimizer with learning rate of order* $O(10^{-4})$ *as well as dropout for the training. Set the input dimension for the generator (i.e., the dimension of the noise vector) in the range between 50 and 200.*

(b) Using the same architectures, train WGAN with Wasserstein loss function. You should change the optimizer to RMSProp and apply parameter clipping in order to enforce the Lipschitz constraint. For RMSProp check ***https://pytorch.org/docs/stable/generated/torch.optim.RMSprop.html*** *while parameter clipping can be enforced via:*

```
for p in discriminator.parameters():
    p.clamp_(-max_amp, max_amp)
```

[1] In case of limited resources, the architectures could have just one conv layer

where 'max_amp' is a hyperparameter that determines the clipping severity. You may explore values in the range between 0.1 and 0.001.

(c) Generate samples from both models and comment on the generated images.

Problem 2 (Conditional Diffusion Model for MNIST digit dataset)**.** *This exercise is an extension of the tutorial on diffusion models* `https://github.com/mikerapt/hy673_tutorials/tree/main/Tutorial-9`.

(a) Modify the code from the tutorial on diffusion models for MNIST digit generation and implement a conditional version of it. Add the conditional one-hot encoding vector as input to the U-Net.

(b) Implement the conditional diffusion model using the classifier-free approach (slide 32 in Lecture 14 and `https://arxiv.org/pdf/2207.12598.pdf`*).*

Problem 3 (Create different avatars from your photo)**.** *Make a photo of yours and go to an online site that generates avatars and create your own avatar. Generate at least five avatars using different conditions regarding appearance, colors, style and backgrounds. Explore the space of avatar making websites (at least two of them) and assess the quality and coherence of the generated avatars for each one.*