Ben-Gurion University of the Negev
Faculty of Engineering Sciences
Department of Software and Information systems Engineering

Deep Learning
Assignment 4

The purpose of the assignment

Enabling students to experiment with building a generative model (GAN) and using it on a real-world dataset. In addition to practical knowledge in the "how to" of building the network, an additional goal is to let students experiment with the problems often associated with GANs: lack of diversity and mode collapse.

Submission instructions:

a) The assignment due date: 11/02/2023
b) The assignment is to be carried out using PyTorch.
c) Submission in **pairs** only. Only **one copy** of the assignment needs to be uploaded to Moodle.
d) Plagiarism of any kind (e.g., GitHub) is forbidden.
e) Submit your work using a notebook.
f) The entire project will be submitted as a single zip file, containing both the report (in PDF form only) and the code. It is the students' responsibility to make sure that the file is valid (i.e. can be opened correctly).

Introduction

In the relevant lecture, we discussed the generative adversarial network (GAN) architecture. The strength of the architecture lies in its ability to generate high quality images (compared to previously-proposed methods) and its lack of overfitting. The weaknesses of the architectures are its inability to ensure diversity, and its tendency to sometimes suffer from mode collapse.

In this assignment you will implement two versions of GAN: a) a "vanilla" GAN (identical to one taught in class), and; b) a conditional GAN (cGAN), which is a simple and empirically effective ways to address the problems mentioned above. The original cGAN paper can be found at this link. You will implement and compare the performance and output of the two architectures.

Instructions

1. Read the above-mentioned paper.
2. Use the MNIST data for your experiments. Because you don't need the test set, you may use the entire datasets for the training of your model.
3. **Part 1:** implement a vanilla GAN on the dataset. Train the model until stability (or until high-quality images are achieved). Produce the following outputs:
    a. Samples of the generated images at the end of each epoch (in case a large number of epochs is required, you may use your judgement and use larger intervals – for example, every two epochs). For each epoch, produce a sample for each digit [0-9].
    b. If your model does not generate images for all digits, specify for which digits it does, and for which digits it doesn't.
    c. Are there specific digits for which the model had a harder time producing (relatively) high-quality images? Which ones? Have you noticed changes in the distribution of the generated images over time (i.e., epochs).
4. Part 2: Modify the vanilla GAN you created in part and turn it into a cGAN. As shown in Figure 1, the required changes are the addition of the desired label (using a one-hot vector) as input to both the generator and the discriminator.
    a. Repeat step 3-a. For each epoch, include one image per digit (because you specify the desired label every time).
    b. Compare the quality of the generated images for each digit. Can you draw conclusions regarding image quality/training speed, etc?

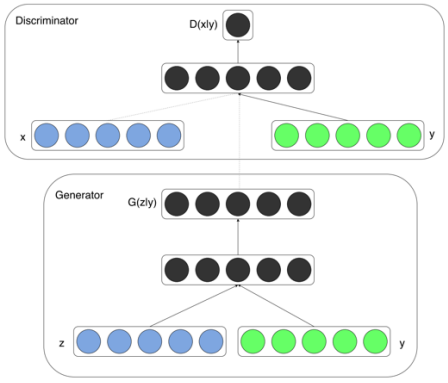

Figure 1: a cGAN architecture