EAI6010 - Module 4

Using an “Off-the-shelf” Model

Submitted to:
Prof. Abhijit Sanyal

Submitted By:
Abhigna Ramamurthy
002982276

Date : 21st June 2022.

Introduction

In this example, Generative and Adversarial Networks for generating computer images is used. Training a generative adversarial network (GAN) to generate new celebrities after showing it pictures of many real celebrities. GANs are a framework for teaching a DL model to capture the training data's distribution so we can generate new data from that same distribution. GANs were invented by Ian Goodfellow in 2014 and first described in the paper `Generative Adversarial Nets. The job of the generator is to spawn 'fake' images that look like the training images. The job of the discriminator is to look at an image and output whether it is a real training image or a fake image from the generator. During training, the generator is constantly trying to outsmart the discriminator by generating better and better fakes, while the discriminator is working to become a better detective and correctly classify the real and fake images. The equilibrium of this game is when the generator is generating perfect fakes that look as if they came directly from the training data, and the discriminator is left to always guess at 50% confidence that the generator output is real or fake. A DCGAN is a direct extension of the GAN described above, except that it explicitly uses convolutional and convolutional-transpose layers in the discriminator and generator, respectively.

Dataset

For this example, I'm using data set of celebrity faces. Using GANs I will try to train the model to generate realistic image or an image that has certain properties for a random input. Two models are deployed to accomplish this. One model that is responsible for taking random input and generating an image with certain characteristics. Another model who objective is to accurately differentiate between images that are generated by the model and images that are in the actual labeled data set (*DCGAN Tutorial*, 2022).

Implementation

With the input parameters set and the dataset prepared, the implementation of GANs can be done. I started with the weight initialization strategy, then the generator, discriminator, loss functions, and training loop. The model weights are randomly assigned from a Normal Distribution with mean as 0 and standard deviation as 0.02. The generator is designed to map latest space vector in dataspace. That is creating a RGB image with the same size as the training images. Discriminator is a binary classification network that takes an image as input and outputs a scalar probability that the input image is real (as opposed to fake). I then created

network instances and setup for training. Trained for 10 epochs to see how the images are generated and got good results.

Results

I investigated three different results. First, we will see how D and G's losses changed during training. Second, we will visualize G's output on the fixed_noise batch for every epoch. And third, we will look at a batch of real data next to a batch of fake data from G. The real and fake images are shown side by side below. Next steps would be to train for longer to see how good the results get. Modifying this model to take a different dataset and possibly change the size of the images and the model architecture.

Figure 1: Model results of real and fake images generated.

References

DCGAN Tutorial. (2022).
GitHub. https://github.com/pytorch/tutorials/blob/master/beginner_source/dcgan_faces_tutorial.py