BME646 and ECE60146: Homework 8

Spring 2023
Due Date: 11:59pm, March 29, 2024
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via BrightSpace. Additional instructions can be found at BrightSpace. Late submissions will be accepted with penalty: **-10 points per-late-day, up to 5 days**.

1 Introduction

The goal of this homework is for you to compare the performance of a GAN and the approach based on diffusion for generative modeling of a dataset. You will be provided with a subset of the famous CelebA dataset. You will write your own code for a GAN to generate fake images using this dataset. About the diffusion based modeling, since it is very compute intensive, you will be provided with a model trained by Aditya Chauhan in RVL and your homework will only involve generating fake images using this model.

If you have the computational resources at your disposal, we will delighted if you train your own diffusion based model using the implementation in the latest addition to DLStudio in Version 2.4.2.

The GAN implementation will only involve the approach based on the DCGAN as explained the Week 11 lecture slides.

This homework will also make you familiar with the Fréchet Inception Distance (FID) to evaluate your generated images **quantitatively**.

2 Getting Ready for This Homework

Before embarking on this homework, do the following:

1. Download DLStudio version 2.4.2 from the following link:

   https://engineering.purdue.edu/kak/distDLS/DLStudio-2.4.2.tar.gz?download

2. You should already be familiar with concept of Transpose Convolution. To review that material, go through Slide 29 through 62 of the Week 8 slide deck on Semantic Segmentation. Make sure you understand

the relationship between the Kernel Size, Padding, and Output Size for Transpose Convolution. You also need to understand the example shown on Slide 48 in which a 4-channel 1×1 noise vector is expanded into a 2-channel 4×4 noise image. Understanding this example is foundational to understanding the working of GAN.

3. Understand the GAN material on Slides 60 through 81 of the Week 11 slide deck. For additional depth, you are encouraged to read the original GAN paper by Goodfellow et al. [1].

4. When you are learning about a new type of a neural network, playing with an implementation by varying its various parameters and seeing how that affects the results can often help you gain deep insights in a short time. If you believe in that philosophy, execute the following the script in the `ExamplesAdversarialLearning` directory of DLStudio:

   ```
   python dcgan_DG1.py
   ```

   It uses the `PurdueShapes5GAN` dataset that is described on Slides 53 through 59 of the Week 11 slides. Instructions for downloading this dataset are on the main DLStudio webpage.

5. Since we are not asking to write your own code for diffusion based modeling, it would be sufficient for you gain an understanding of just the top-level ideas on Slides 123 through 165. Ask yourself the following questions: 1) Why does diffusion modeling require two Markov Chains? 2) What is the difference between the forward q-chain and the reverse p-chain? Why does injecting Gaussian noise make it easier to train a diffusion based data modeler? etc.

6. Make yourself with how to run the diffusion related code in DLStudio. To that end, go over the following page to understand how that code is organized:

   https://engineering.purdue.edu/kak/distDLS/GenerativeDiffusion-2.4.2_CodeOnly.html

7. After you have downloaded and installed Version 2.4.2 of DLStudio, read the README in the `ExamplesDiffusion` directory for how to run the diffusion code in DLStudio.

3 Programming Tasks — GAN

1. Ensure you have downloaded the subset of the CelebA dataset from BrightSpace that is associated with this homework. The dataset comprises 10,000 images for working with your two data modelers. All these images are of size 64×64. Figure 1 shows a sample of the images from the dataset.

2. Design your own generator and discriminator networks. Just like the previous homeworks, you have total freedom on how you design your networks. You can draw inspiration from the implementations in DLStudio.

   Your generator must be able to generate RGB celebrity images of size 64×64 from random noise vectors utilizing *transpose convolutions*.

3. Subsequently, you'll need to write your own adversarial training logic. You can refer to Slide 64 through 69 of the Week 11 slides to familiarize yourself with how it can be done. You only need to use the nn.BCELoss for training.

4. **In your report, plot the adversarial losses over training iterations for both the generator and the discriminator in the same figure.**

4 The Task Related to Diffusion

You will find the following scripts in the directory ExamplesDiffusion:

1. README
2. RunCodeForDiffusion.py
3. GenerateNewImageSamples.py
4. VisualizeSamples.py

You may want to start with reading the README file.

The following instructions are to generate images using the network weights provided to you.

Figure 1: Sample images from the subset of CelebA dataset provided.

1. As you have already read, you will need to run all three files to see diffusion from end-to-end. However, we understand that one may not have sufficient computation capacity to run multiple epochs of Diffusion training for batch size 32. Therefore, we are providing the weights to you and you may skip running `RunCodeForDiffusion.py`.

2. To generate images using the pretrained diffusion model, first change the results directory to your downloaded path directory of weights in `GenerateNewImageSamples.py` and run the file.

   NOTE: Ensure you have installed the new version of DLStudio 2.4.2 before running this code.

3. Generate 1000 fake images. This will create npy files which you can visualize using the `VisualizeSamples.py`. Make sure you also change the directory locations accordingly in the `VisualizeSamples.py` file as well.

If you wish to train your own diffusion model, you will also need to run the `RunCodeForDiffusion.py` before executing the generation and visualization scripts. Change the directory locations in the `RunCodeForDiffusion.py` before executing the file.

4.1 Evaluating Your GAN

You can visually analyze the outputs generated by your face-generator. However, how does one quantitatively evaluate generated images? For evaluating generated images quantitatively, Fréchet Inception Distance (FID) is used. Originally proposed in [2], the FID is a widely used metrics for measuring both the quality and the diversity of GAN-generated images. More specifically, it does so by measuring how close the distribution of the fake images is to the distribution of the real images.

1. First, you should generate 1k images of fake images from randomly sampled noise vectors using your trained generator BCE-GAN.

2. To calculate the FID, one would first encode the set of real images (from training data) into feature vectors using a pre-trained Inception network, and then model the resulting distribution of feature vectors using a multivariate Gaussian distribution. The same is carried out for the set of fake images. Once that is done, the FID is simply the Fréchet distance between the two multivariate Gaussian distributions.

3. For this homework, you will be using the `pytorch-fid` package [3] for calculating the FIDs. To install the package, use the command:

   ```
   pip install pytorch-fid
   ```

   Once installed, you can use the `pytorch-fid` package in a Python script as follows:

```python
from pytorch_fid.fid_score \
    import calculate_activation_statistics, \
    calculate_frechet_distance
from pytorch_fid.inception import InceptionV3

# you have to write a script to populate the following
                                path lists
real_paths = ['/real/0.jpg', '/real/1.jpg', ...]
fake_paths = ['/fake/0.jpg', '/fake/1.jpg', ...]
dims = 2048
block_idx = InceptionV3.BLOCK_INDEX_BY_DIM[dims]
model = InceptionV3([block_idx]).to(device)
m1, s1 = calculate_activation_statistics(
    real_paths, model, device=device)
m2, s2 = calculate_activation_statistics(
    fake_paths, model, device=device)
fid_value = calculate_frechet_distance(m1, s1, m2, s2)
print(f'FID: {fid_value:.2f}')
```

In your report, you will have to present both qualitative and quantitative results:

4.
   - **Qualitative Evaluation**: Display a 4×4 image grid, similar to what is shown in Figure 1, showcasing images randomly generated by your BCE-GAN. Repeat the same with images generated by your Diffusion output. Describe in several lines what are the differences and similarities in the respective outputs
   - **Quantitative Evaluation**: Present the FID values for both GAN and Diffusion variants.

   Finally, include a paragraph discussing your results: BCE-GAN v.s. Diffusion, which is better?

5 Submission Instructions

Include a typed report explaining how did you solve the given programming tasks.

1. Your pdf must include a description of
   - The figures and descriptions as mentioned in Sec. 3.
   - Your source code. Make sure that your source code files are adequately commented and cleaned up.
2. Turn in a pdf file a typed self-contained report with source code and results. Rename your .pdf file as hw8_<First Name><Last Name>.pdf
3. Turn in a zipped file, it should include all source code files (only .py files are accepted). Rename your .zip file as hw8_<First Name><Last Name>.zip .
4. **Make sure your submission zip file is under 10MB.** Compress your figures if needed.
5. **Do NOT submit your network weights.**
6. **Do NOT submit your dataset.**
7. For all homeworks, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert it to `.py` and submit that as source code.

8. You can resubmit a homework assignment as many times as you want up to the deadline. Each submission will overwrite any previous submission. **If you are submitting late, do it only once on BrightSpace.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.

9. The sample solutions from previous years are for reference only. **Your code and final report must be your own work.**

10. To help better provide feedback to you, make sure to **number your figures**.

References

[1] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

[2] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[3] Maximilian Seitzer. pytorch-fid: FID Score for PyTorch. https://github.com/mseitzer/pytorch-fid, August 2020. Version 0.3.0.