BME646 and ECE60146: Homework 9

Spring 2025
Due Date: 11:59pm, April 7, 2025
TA: Akshita Kamsali (akamsali@purdue.edu)

Turn in typed solutions via Gradescope. Post questions to Piazza. Additional instructions can be found at the end. Late submissions will be accepted with penalty: -10 points per-late-day, up to 5 days.

1 Special Notes

- Except for one important difference, this homework is largely the same as the one given last year on the subject of generative data modeling. The difference is in Section 3.4 where you are being asked to compare directly the two different generative methods, GAN and Diffusion.
- You are once again being given extended time for the homework. To provide you with an incentive to not postpone the homework until it is too close to the deadline, you will also have to honor an early-proof-of-progress deadline, which is going to be 11:59pm, April 2. You will submit a brief report on the progress made on the homework between now and then. **Your final homework solution will NOT be graded if it turns out that you did not submit the progress report by the April 2 deadline.**

2 Introduction

Generative models have revolutionized how we create realistic images, from deepfake technology to AI-generated artwork. But not all generative models work the same way. Two of the most powerful approaches — Generative Adversarial Networks (GANs) and Diffusion Models — have emerged as leading contenders. However, they have fundamental differences in how they generate images and capture data distributions.

In this homework, you will explore these two paradigms and compare their effectiveness with a subset of the CelebA dataset. Through hands-on experimentation, you will gain an intuitive understanding of how each model learns, where they succeed, and where they struggle.

3 Getting Ready for This Homework

Before embarking on this homework, do the following:

1. You are already 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.

2. 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].

3. 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.

4. Since we are not asking you 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 125 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.

5. 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-2.5.1_CodeOnly.html

6. After you have downloaded and installed Version 2.5.1 of DLStudio, read the README in the `ExamplesDiffusion` directory for how to run the diffusion code in DLStudio.

4 Programming Tasks

Before you begin, download the supplementary material under HW9 from Brightspace. This supplementary folder has a subset of CelebA dataset with 10,000 images and weights for Diffusion model trained on CelebA dataset. All these images in the subset of the CelebA dataset are of size 64×64. Figure 1 shows a sample of the images from the dataset.

4.1 Train GAN

1. Drawing inspiration from DLStudio to design your own generator and discriminator networks. Just like the previous homeworks, you have total freedom on how you design your networks.

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

2. 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.

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

4.2 Run Diffusion

You will find the following scripts in the directory ExamplesDiffusion:

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

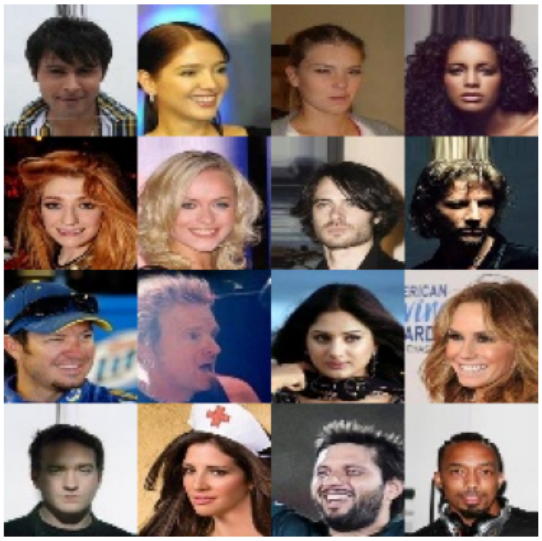

Figure 1: Sample images from the subset of CelebA dataset provided.

You may want to start with reading the README file.

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

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 you may not have sufficient computing 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.

3. Generate 1024 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. Please note this code is for visulization. You may have to modify the code for the Section 4.3.

If you wish to train your own diffusion model, you will also need to run the `RunCodeForDiffusion.py` before executing the generation and visual-

Figure 2: Images generated with 500 gaussian steps with the provided weights

ization scripts. Change the directory locations in the `RunCodeForDiffusion.py` before executing the file.

4.3 Evaluation: Frechet Inception Distance

You can visually analyze the outputs generated by your face-generator. However, how does one quantitatively evaluate generated images? For evaluating generated images quantitatively, Frechet 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 1024 images of fake images from randomly sampled noise vectors using your trained generator GAN.

2. Display 4x4 (16 images) images generated by GAN and Diffusion Model each.

3. 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 Frchet distance between the two multivariate Gaussian distributions.

4. 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:

5. - **Qualitative Evaluation**: Display a 4×4 image grid, similar to what is shown in Figure 1, showcasing images randomly generated by your 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. Visually inspect the 1024 images you generated with DCGAN and the Diffusion model. Identify signs of mode collapse by checking whether the GAN generates the same gibberish pattern in all the batch instances and for all the batches.
   - **Quantitative Evaluation**: Present the FID values for both GAN and Diffusion variants.

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

4.4 Comparing the Two Generative Methods

1. Generate images using a pre-trained diffusion model weights provided to you.
2. Use the diffusion-generated images as additional training data for fine-tuning a DCGAN. Fine-tuning means taking the weights of your DCGAN at the last epoch you had and then restarting the training using the diffusion-generated images. You may use the same hyperparameters or experiment with different ones.
3. You may use the 1024 images you generated for FID calculation.
4. Compare the visual quality and diversity of images generated by the standard GAN, the fine-tuned GAN and Diffusion based model.
5. Display 4x4 (16 images) images generated by fine-tuned GAN
6. Discuss how training with diffusion-generated images affects the GANs performance.

5 Submission Instructions

Include a typed report explaining how you solved the given programming tasks. You may refer to the homework solutions posted at the class website for the previous years for examples of how to structure your report

1. **Turn in a PDF file and mark all pages on gradescope**.
2. Submit your code files(s) as zip file.
3. **Code and Output Placement:** Include the output directly next to the corresponding code block in your submission. Avoid placing the code and output in separate sections as this can make it difficult to follow.
4. **Output Requirement:** Ensure that all your code produces outputs and that these outputs are included in the submitted PDF. Submissions without outputs may not receive full credit, even if the code appears correct.

5. For this homework, you are encouraged to use `.ipynb` for development and the report. If you use `.ipynb`, please convert code to `.py` and submit that as source code. **Do NOT submit .ipynb notebooks.**

6. You can resubmit a homework homework 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.** Otherwise, we cannot guarantee that your latest submission will be pulled for grading and will not accept related regrade requests.

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

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.