# ECE60146 Homework 8

Utsav A. Negi

March 27, 2024

## Aim

The aim of this report is to demonstrate the approach used to implement image modelling using Generative Adversarial Nets and Denoising Diffusion using a subset of CelebA dataset.

### Source Code Detail

For this assignment, the details about the source code are as follows:

* *hw8.py*: Source code consisting of all the essential classes, functions & main ML pipeline for training and evaluating GANs along with code to evaluate the samples generated using Denoising Diffusion.

### Image Modelling using Generative Adversarial Nets (GAN)

### Background

This implementation consists of two nets: Discriminator and Generator. The role of the discriminator is to return the probability that the input belongs to the distribution representing the training data. The role of the Generator is to create images from noisy data by transforming the noisy data distribution to a distribution representing the training data. For this assignment, the Discriminator and Generator architecture is like the DLStudio implementation, while the number of input-output channels and the number of layers is tuned to convert the CelebA image data into its latent form for the Discriminator using Conv2D and vice versa for the Generator using ConvTranspose2D.

### Training

For training the nets, weights for convolutional and batch normalization layers are initialized to stabilize the nets. Two separate ADAM optimizers with the same learning rate and hyperparameters are used to determine the new parameters for the Discriminator and the Generator. To calculate the loss, the criterion is set to BinaryCrossEntropy Loss Function as the values are returned in the form of a probability. In each iteration, the outputs of the Discriminator net are compared with the labels filled with 1 to determine the loss based on the training images. At the same time, the output image of the Generator net, generated from a Gaussian noise, is used as an input to the Discriminator net. The output values yielded by the Discriminator are compared with the labels filled with 0 to determine the loss based on the fake images. These losses are backpropagated and added to determine the new parameters for the Discriminator net. Using the fake image generated by the Generator net, the newly trained Discriminator net generates an output which is compared with the labels filled with 1 to determine the loss used to calculate the new parameters for the Generator net. In this implementation, the number of epochs is set to 500 while the learning rate and the β for both optimizers are set to 1e-4 and (0.75, 0.999) respectively.

### Observations

A graph of blue and orange lines

Description automatically generated

*Figure 1: Training Loss of Discriminator and Generator with every iteration.*

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Figure 2: Images generated by the generator after training completion.*

From Figure 1, it can be observed that with each iteration, the generator loss is increasing while the discriminator loss is decreasing at a slower rate. Using the trained generator model, 2048 images are sampled, which are like the images shown in Figure 2. These 2048 generated images are compared with 2048 real images by determining the Frechet Inception Distance which was around 145. From Figure 2, it can be inferred that the model generates similar images with same hairstyle and face structure, yet samples lack clarity in defining the facial features.

### Image Modelling using Denoising Diffusion

### Background and Training

The method of Denoising Diffusion involves two Markov chain processes taking place simultaneously. One process called Diffusion takes place by injecting Gaussian noise into a training image at each timestep until the image data gets substituted with an isotropic Gaussian noise. Another process called Denoising takes place by removing Gaussian noise from a sample of isotropic Gaussian noise at each timestep until the sample gets converted into a recognizable image. During these processes, a denoising neural network is trained to remove the same amount of noise which was added during the diffusion for the same timestep transition.

For this assignment, pre-trained weights are added to the Unet model to generate images due to hardware constraints. The number of timesteps are set to 100 and around 2048 sample images are generated for evaluating the performance.

### Observations

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

*Figure 3: Images generated using Denoising Diffusion.*

Using the trained Unet model, 2048 images are sampled, which are like the images shown in Figure 3. These 2048 generated images are compared with 2048 real images by determining the Frechet Inception Distance which was around 131. From Figure 3, it can be inferred that images have sharp facial features and face structure; however, some samples struggle in getting hair correctly.

### Verdict between Generative Adversarial Nets & Denoising Diffusion

The images generated by GAN consists of noisy facial features compared to the images generated using Denoising Diffusion. However, the images generated by Denoising Diffusion are soft and lack hair features. Based on FID values and the results, it is clear that the Denoising Diffusion method is better at generating images compared to GAN. This can be because GAN are sensitive and any changes in hyperparameters or weights can have a huge impact on the generated images. However, with better hyperparameters and model architecture, GAN generated images can compete with the images generated using the Denoising Diffusion.