FACE GENERATOR USING DEEP GENERATIVE ADVERSARIAL NETWORKS
Individual Report

Columbian College of Arts & Sciences
The George Washington University, Washington D.C.

Student:
Aira Domingo
Professor: Amir Jafari
Date: April 2020

1. Introduction

In today's society, much of the news that we receive is through the internet and social media. Many individuals form their decisions and opinions based on what they see online. The danger in this is that not everything we see online is real and truthful. There are a lot of manipulated data and fake media circulating the internet that we do not realize. Machine learning has played a huge role in the creation of many fake images and videos. One technique used nowadays is a Generative Adversarial Network (GAN).

GANs, developed by Ian Goodfellow, is a two-player network that compete against each other. The first network is a generator that tries to imitate the distribution of the given data. The second is a discriminator which estimates the probability that a sample is from the given data rather than the data produced by the generator. The two models are trained at the same time and learn from each other. The two networks are both differentiable, so they are able to optimize using the same loss function. While the discriminator's goal is to maximize this loss function, the generator's objective is to minimize it.

For this project, we utilized the Deep Convolutional Generative Adversarial Network to generate fake faces.

2. Description of Individual Work

Much of the work needed to prepare for this project was research. A lot of time was dedicated to understanding the architecture of Generative Adversarial Networks. Renzo and I both took the time to understand the overall process of how such realistic images are created by GANs. After reading the original GAN paper by Ian Goodfellow and a few other resources, I was able to understand the loss function below.

$$\min_G \max_D V(D, G) = \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

In GANs, the discriminator is trained with mini batch stochastic gradient ascent on this loss function and the generator is trained with mini batch stochastic gradient descent. The discriminator wants to maximize this function which would mean it is correctly identifying if the sample is from the real given data or it is from the fake generated data. Since only the second half of this function is dependent on the generator, the generator only needs to optimize this second half. By minimizing log(1-D(G(Z))), it is also maximizing the probability that the discriminator is incorrectly identifying a generated image as real (D(G(z))).

In training the model, I tried different numbers of layers, image size, parameters, and researched the impact of batch size and learning rate on the model. Each time I trained the model, I named it a different pickle file and compiled my results. An example of my results are shown below.

3. Description of Portion of Work

After agreeing with which dataset to use, I created the load.py file to download and unzip the dataset from the Large-scale CelebFaces Attributes (CelebA) Dataset Google Drive to our virtual machines.

The dataloader function takes the training data in batches and preprocesses it. The preprocessing includes cropping the images in the center and resizing it to a given size, which was initially 32. The data was also converted to a tensor in this process. Next, since the generator uses a Tanh activation function in its output layer, the loaded training images' must also be in the same range, which is from -1 to 1.

The generator takes in random noise from a Gaussian distribution and goes to a Linear layer where it is reshaped into a 4-dimensional tensor. This is then followed by 4 transposed convolution layers where upsampling is done and an image is created from 4x4, to 8x8, 16x16, and finally outputs 32x32. I used Radford's paper to guide the creation of the model. Each transposed convolutional layer included a batch normalization except for the last layer. This helps with the training problems that arise from poor initialization since it prevents the generator from collapsing all the samples to a single point. The hidden layers in the generator used ReLu as an activation function because having a lower bound of 0 supposedly helped the model learn the color space of the given data's distribution faster.

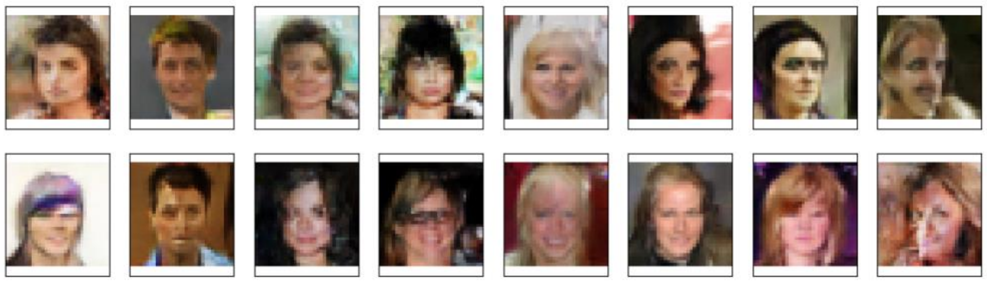
For the project's presentation, Renzo and I split up the slides that we would talk about. I presented the introduction, discussed the network's loss function, and the results.

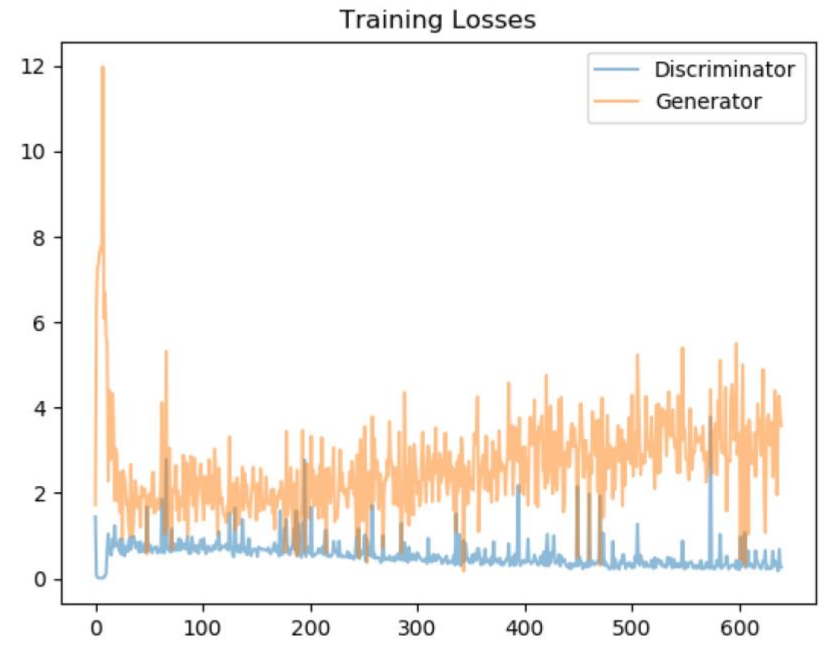
For the final report, Renzo and I also split up the sections of the paper. I introduced the GAN and DCGAN in the introduction. In the experimental set up portion, I discussed the loading and preprocessing of the data, the generator, and the loss function. I also discussed the results. Renzo and I both made contributions to the summary and conclusion. We also went through the rest of the other sections and added details where necessary.

4. Results

Our generator was indeed able to produce face images, although the quality was not the best. We observed that the training of this model was very time consuming. Running for 40 epochs took about 4 hours. We also noticed in the training losses that the generator's loss was increasing, and the discriminator's loss was decreasing, which was inconsistent with the objective of GANs. The discriminator had the upper hand which is often a problem associated with training GANs. Another problem we came across was mode collapse. In certain trials of training, the generator produced similar images. In future research, we could try using Wasserstein loss to solve this problem.

This is the result of setting the batch size = 256, epochs = 40, learning rate = 0.0002, and having the batch normalization at every layer. I observed that some images look similar, which is the result of mode collapse.





Additionally, I believe that by using a higher quality dataset in future trainings of this network and training for a longer period of time, we would be able to create a more complex model that would generate more realistic looking images.

5. Summary and Conclusions

In our society today, much of our time is spent on the internet where we are constantly accessing different types of data and information. Unfortunately, there are also fake data and media circulated around that have led others to make uninformed decisions based on this manipulated data. This has become possible through advancements in technology, such as Generative Adversarial Networks.

This project explores a flavor of GANs called Deep Convolutional Generative Adversarial Networks (DCGANs) to create face images and involves two networks in a minimax game. The DCGAN includes a generator which generates images and a discriminator which distinguishes real images from fake ones produced by the generator. Both networks are trained with the same loss function. Mini batch stochastic gradient descent is used to train the generator and mini batch stochastic gradient is used to train the discriminator.

The generator was successful in creating face images. However, more work needs to be done to improve the quality of the output images. Training with a high-quality dataset for a long time period could allow us to build a more complex model to generate very realistic face images that other people would not recognize as fake.

After completing this project, we have seen how feasible it is to create fake data. With this in mind, better technology must be developed to be able to identify manipulated data in order to mitigate the spread of false information.

6. Percentage of code calculation

$$\frac{96 - 24}{96 + \ 41} \ x\ 100 = 52.55\%$$

References

Brownlee, J. (2019, July 12). A Tour of Generative Adversarial Network Models. Retrieved from https://machinelearningmastery.com/tour-of-generative-adversarial-network-models/

Brownlee, J. (2019, November 21). How to Explore the GAN Latent Space When Generating Faces. Retrieved from https://machinelearningmastery.com/how-to-interpolate-and-perform-vector-arithmetic-with-faces-using-a-generative-adversarial-network/

Common Problems | Generative Adversarial Networks | Google Developers. (n.d.). Retrieved from https://developers.google.com/machine-learning/gan/problems

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., … Bengio, Y. (2014). Generative Adversarial Nets. Retrieved from https://arxiv.org/pdf/1406.2661.pdf

Hui, J. (2019, October 29). GAN - Why it is so hard to train Generative Adversarial Networks! Retrieved from https://medium.com/@jonathan_hui/gan-why-it-is-so-hard-to-train-generative-advisory-networks-819a86b3750b

Radford, A., Metz, L., & Chintala, S. (2016, January 7). UNSUPERVISED REPRESENTATION LEARNING WITH DEEP CONVOLUTIONAL GENERATIVE ADVERSARIAL NETWORKS. Retrieved from https://arxiv.org/pdf/1511.06434.pdf

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., & Chen, X. (2016, June 10). Improved Techniques for Training GANs. Retrieved from https://arxiv.org/abs/1606.03498

Shen, K. (2018, June 19). Effect of batch size on training dynamics. Retrieved from https://medium.com/mini-distill/effect-of-batch-size-on-training-dynamics-21c14f7a716e

Shibuya, N. (2019, January 1). Up-sampling with Transposed Convolution. Retrieved from https://medium.com/activating-robotic-minds/up-sampling-with-transposed-convolution-9ae4f2df52d0

Tiu, E. (2020, February 4). Understanding Latent Space in Machine Learning. Retrieved from https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d

Uniqtech. (2019, June 13). Multilayer Perceptron (MLP) vs Convolutional Neural Network in Deep Learning. Retrieved from https://medium.com/data-science-bootcamp/multilayer-perceptron-mlp-vs-convolutional-neural-network-in-deep-learning-c890f487a8f1

Vijay, R. (2019, November 12). Fake Face Generator Using DCGAN Model. Retrieved from https://towardsdatascience.com/fake-face-generator-using-dcgan-model-ae9322ccfd65