FACE GENERATOR USING DEEP GENERATIVE ADVERSARIAL NETWORKS
Individual Report

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

Student:
Renzo Castagnino
Professor: Amir Jafari
Date: April 2020

1. Introduction

Machine learning is changing the world we live. Huge amount of data makes it possible to create better and solid models. While traditional machine learning models are still being used, there are new ones being improved and created. That is the case for a new a model called Generative Adversarial Networks (GANs) proposed by Ian Goodfellow in 2014. Basically, this unsupervised model consists of two neural networks, the discriminator and the generator, that compete with each other. For instance, this model can be trained on photographs and can generate new images that look as authentic to human eye. This model can have many applications, for instance StyleGAN, a state-of-the-art model introduced by Nvidia, can generate a face images that looks very realistic and they could be confused as a real person.

While many concerns about malicious applications on with this technology such as fake images, or manipulated videos, the GAN concept still remains very interesting and worth of investigation.

2. Description of individual work

The main purpose of the project is to apply the GAN concept, to create fake human faces that can look as realistic as possible. In order to this, we needed to understand the underlying concept of GANs. The project was about 75% research and 25% working on preprocessing and modeling and getting the results.

It is very important to know the theory behind the application, and thus me an Aira both research about the principles of GAN. However, we divided a deeper research: for the discriminator, I was in charge of understanding the overall architecture and best practices, while Aira worked in understanding the generator.

In regards of the Discriminator, it could be considered as a function that maps images to a probability of a real distribution D: D(x) → (0, 1). While having the generator G fixed, the discriminator performs gradient ascent and is trained to classify images as either being from the original data, or from the latent space Z.

Furthermore, as part of my work, it was important to understand how the model was going to be trained and learn. Thus, I needed to understand how the loss function works. In the case of the discriminator, we used stochastic gradient ascend:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right]$$

Gradient ascend is used to deal with the vanishing gradient. Considering this the total loss for the discriminator is the sum of the real image's loss, plus the loss for the fake images. For our GAN we defined two optimizers, one for the generator and another one for the discriminator. In our case, we used Adam as an optimizer.

3. Description of portion of work

As part of the process, and once we had the images loaded in the cloud, we wanted to visualize the original images to have an idea of their quality. For this, I created a function that iterates over the train_loader previously created in PyTorch, and that plots 20 sample images using Pyplot.

After that, I worked on the discriminator part. I used 3 convolutional layers and one fully connected layer at the end that outputs a logit. All the layers except the first, have a batch normalization. According to my research, by normalizing the input to have zero mean and unit variance, allow us to create deeper layers. I used Leaky ReLU as activation function. This allow the model to avoid saturation in the gradients thanks to its lower bound slightly lower than zero.

While it is common to use max-pooling layers for downsampling, in GAN is recommended to use strided convolutions, and I use a value stride of 2 for the layers, with a padding of 1. The output of each layer will become half of the original input, this means that the input was 32x32, but then it went to 16, 8, 4.

After that, we can calculate our loss function. According to the research, we use the Binary Cross-Entropy Loss with logit loss, because is considered a more stable version of the normal BCE. In terms of the initialization, to make the model converge, the weights were initialized with a normal distribution zero mean and a standard deviation of 0.002. For the optimizers, we start with a learning rate of 0.002, but after researching, we found that a good starting point was 0.0002. Also, we found that using a momentum $ß_1$ of 0.001 would make the training oscillate, so we reduce it to 0.5 to stabilize the model.

After the training, the images were saved on a pickle file to be explored. To do this, I created a different file to load the pickle, and plot the resulting images with Pyplot.

Finally, for the project requirements, both Aira and I worked on the presentation and the project report.

4. Results

While our results were not the best in terms of quality, I can say that the model was able to create fake faces. While changing the hyperparameters allowed us to slightly improve the

model, we could have improved the results by increasing the input in the image for instance to 128x128 pixels and add more hidden layers to get better results. Also, the timing to train the mode was on average 4 hours. Increasing the number of epochs and hidden layers could have given us better results, but also it would have been time-consuming.

5. Summary and conclusions

While in recent years supervised learning with CNN has had many applications in computer vision, unsupervised learning DCGAN is a new model worth of study. This project has allowed me to understand the theory behind GANs. Initially, I was expecting that DCGANs to be something completely different. However, the principles remain the same as discussed during the classes. There are a couple of neural networks, you have an input a batch of real images, and fake images that comes from a random vector, you have to perform the feedforward, and also backpropagation (for both the discriminator and the generator).

Having a different version of a typical loss function (for example, usually you'll do stochastic gradient descend, but for the discriminator you need to do stochastic gradient ascend), made me understand the importance of having a good foundation on the concepts. In order to understand the training process, I needed to understand specifically how the loss function works, for which I had to go deeper in topics such as binary cross entropy and maximum likelihood estimation, the importance of batch normalization, difference between ReLU and Leaky ReLU (due to vanishing gradient).

Finally, I can conclude that the project was successful. Although the results were not the best in terms of the quality of fake images generated, I was able to understand the main concept behind GANs. I would continue research more about this topic, as well as other new methods such as progressive growing or other techniques that can give better results.

6. Percentage of code calculation

$$\frac{96 - 22}{96 + 53} x\ 100 = 49.66\%$$

7. References

1. AlShariah1, N. M. (n.d.). Detecting Fake Images on Social Media using Machine Learning. Retrieved from https://thesai.org/Downloads/Volume10No12/Paper_24-Detecting_Fake_Images_on_Social_Media.pdf

2. 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/

3. Creswell§, A., Dumoulin, V., & Arulkumaran, K. (2017, October 19). Generative Adversarial Networks: An Overview. Retrieved from https://arxiv.org/pdf/1710.07035.pdf

4. Dinesh. (2019, November 28). CNN vs MLP for Image Classification. Retrieved from https://medium.com/analytics-vidhya/cnn-convolutional-neural-network-8d0a292b4498

5. 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

6. Kingma, D., & Ba, J. L. (2017, January 30). ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. Retrieved from https://arxiv.org/pdf/1412.6980.pdf

7. Liu, Z., & Luo, P. (2016, August 7). Large-scale CelebFaces Attributes (CelebA) Dataset. Retrieved from http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

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

9. Silva, T. S. (2018, August 11). Advanced GANs - Exploring Normalization Techniques for GAN training: Self-Attention and Spectral Norm. Retrieved from https://sthalles.github.io/advanced_gans/

10. Vincent, J. (2018, December 17). These faces show how far AI image generation has advanced in just four years. Retrieved from https://www.theverge.com/2018/12/17/18144356/ai-image-generation-fake-faces-people-nvidia-generative-adversarial-networks-gans

11. VGGFACE2. (n.d.). Retrieved from http://www.robots.ox.ac.uk/~vgg/data/vgg_face2/data_infor.html

12. Xiang1, S., & Li1, H. (2017, December 4). On the Effects of Batch and Weight Normalization in Generative Adversarial Networks. Retrieved from https://arxiv.org/pdf/1704.03971.pdf

13. Sudhir, K. (2017, June 22). Generative Adversarial Networks- History and Overview. Retrieved from https://towardsdatascience.com/generative-adversarial-networks-history-and-overview-7effbb713545

14. Sudhir, K. (2017, June 22). Generative Adversarial Networks- History and Overview. Retrieved from https://towardsdatascience.com/generative-adversarial-networks-history-and-overview-7effbb713545