Cool GANs & Fake Celebrities

Project in DD2424 Deep Learning in Data Science

Diogo Pinheiro, Jakob Lindén, Márk Csizmadia, Patrick Jonsson
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology
May 25, 2021

Abstract

When it comes to recent exciting innovations in the Machine learning field, Generative Adverbial Networks (GANs) are on top of the list. The ability to generate new data instances with resemblance to the training data has caught the attention of many researchers. In this project we present one of its extensions, Deep Convolutional GAN (DCGAN). Here, the main goal is to generate human faces by providing as training data a dataset containing faces of celebrities. Additionally, we experiment and compare results with the guidelines provided in the original paper by looking at the Fréchet Inception Distance (FID).

1 Introduction

In this project we explore the potential of using a Deep Convolutional Generative Adversarial Network (DCGAN) to generate high quality images of fake human faces. Being able to generate new data is one of the benefits of using a generative model over a discriminant model. It also learns the joint distribution of the data and is able to tell how probable a sample is. GANs are not the only type of generative models, there are other methods such as Variational Autoencoders (VAE) that could be used.

The VAE however uses the mean squared error which is an imperfect criteria to optimize the image pixel-wise, this averaging combined with noise tends to result in blurry images [3]. The aforementioned upsides of using GANs however does come with a much more difficult training process compared to discriminant models.

Interest in the field of computer vision is and has been increasing for several years now, and to be able to handle more demanding image related tasks we not only need good hardware but also advanced software to be able to execute these tasks. Generative models such as GANs has already proven to be useful in several areas of computer vision some of these being: creating super-resolution images from low resolution images, video predictions, image to image translation and text to image translation.

Here, we compared the suggestions provided in the original paper to define a model that allowed the best performance for the 32x32 images, by not only evaluating the quality of the generated images but also looking at a quantitative measure, the FID score. The resulting images present a clear illustration of a face, with a FID score of 102.

2 Data

To train the network we use a large data set containing images of celebrities. This data set is known as the CelebA data set [4]. The data set consists of 202,599 images of faces from 10,177 different identities. The images in the data set has been obtain from the internet for non commercial research purposes. It contains 40 different features such as: wearing glasses, wavy hair, male, big nose, high cheekbone, receding hairline etc. It also contains 5 different landmark locations describing the position of the eyes, nose and mouth in the image of a face.

Before the training process begins the data is pre-processed by rescaling the values to be in the interval from -1 to 1. The images are also reshaped from an original dimension of (218, 178, 3) to (32, 32, 3). Examples of the resulting images after this rescaling and reshaping are shown in Figure 1.

The reshaping to fewer pixels is a way to make the training process easier at the cost of image quality. This is done due to the fact that training the DCGAN is a very computationally heavy task.

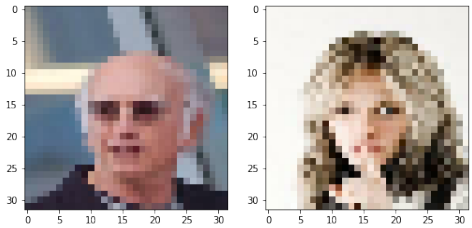


Figure 1: Examples of rescaled and reshaped images.

3 Method

The Generative Adversarial Network (GAN) consists of two major parts, the generator and the discriminator. These are setup in a competitive dynamic where the discriminator is trying to distinguish between fake samples generated by the generator and real samples from the data. The generator is therefore trying to generate fake samples as close to the real samples as possible to make the job as hard as possible for the discriminator part of the network. This creates a dynamic where both parts of the network improves the other, which is the source to the powerful performance of GANs [1]. The discriminator and generator are both defined by multilayer perceptrons and the network is trained using backpropogation.

In this project we focus on implementing an extension of the GAN called DCGAN. The idea behind this is to use a more powerful network architecture by replacing the multilayer perceptrons with convolutional neural networks (CNNs) [5].

The CNN consists of convolutional layers which in turn consists of a filter (or filters) which slide across the input image to examine the image in smaller parts. The filter sliding across the input image where the number of pixels the filter moves at a time is called stride, this outputs a response map, the response map is in this way the data that gets feed into the next layer if there are multiple convolutional layers in the network. This process makes it possible for the CNN to examine features of the images and finally distinguish different objects in the images [6].

The architecture for the DCGAN generator is visualized in Figure 2, the discriminator is not visualized since it is just a horizontal flip of the generator architecture. Additionally in the discriminator strided convolutions are used while in the generator fractional-strided convolutions are used. Fractional-strided convolutions is the act of inserting zeros between input units, which causes the filter to move around the image at a lower speed. Batch normalization is applied to each layer except the output of the generator and the input of the discriminator. Furthermore *LeakyReLU* is used as an activation function for layers except the output of the generator which uses *tanh* as activation function.

Similar to the generator's architecture, the discriminator can be visualized as a horizontally flipped version of the generator in Figure 2.

The process of then training the DCGAN is done by using mini-batch gradient descent with a batch size of 128, together with the Adam optimizer. As for the parameter settings of the optimizer [5] found that setting the learning rate to 0.0002 and the momentum term β_1 to 0.5 to be stable for training, therefore we use the same settings in this project.

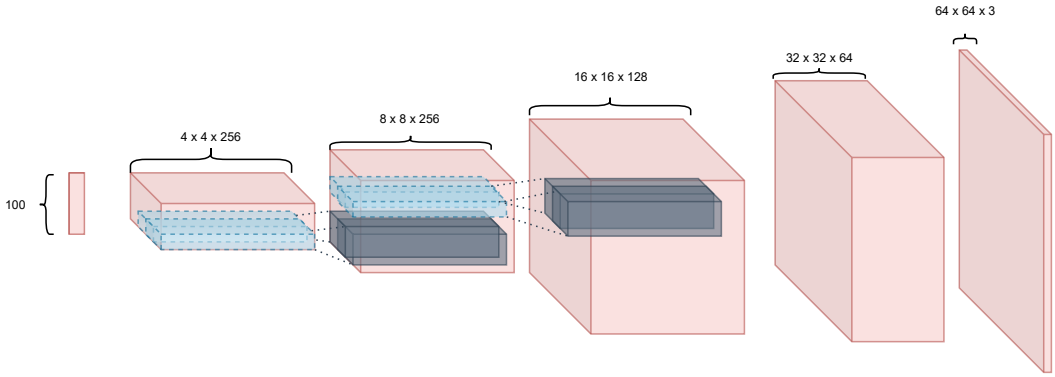


Figure 2: DCGAN generator architecture.

For model comparison and evaluation the Fréchet Inception Distance (FID) is introduced as a metric. FID is a distance measure that is used to compare the images that were generated by the models generator part and the real images in the dataset. This is done by feeding the images through an Inception v3 model that computes computer vision statistics adequate for comparison. More precisely, two Gaussians, one representing the real images and one representing the generated images are compared. The distance d is measured between (m, C) and (m_w, C_w) which is the mean and the covariance corresponding to the Gaussians of the real images and the generated images. [2] The formula to calculate the FID is then defined as:

$$d^2\left((m, C), (m_w, C_w)\right) = \|m - m_w\|_2^2 + Tr\left(C + C_w - 2\left(CC_w\right)^{1/2}\right) \tag{1}$$

4 Experiments

First and foremost, in order to check that our DCGAN model is working as expected we started by applying it to a commonly used dataset, MNIST. Here, the complexity is quite small when compared to the main goal of the project, but nevertheless the results observed in Figure 3 show great potential for the model. The generated digit, shown as example in the second figure, doesn't have any clear difference to the original dataset.

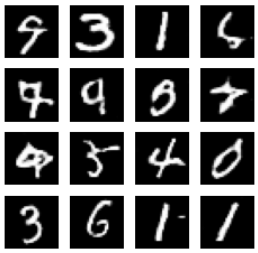


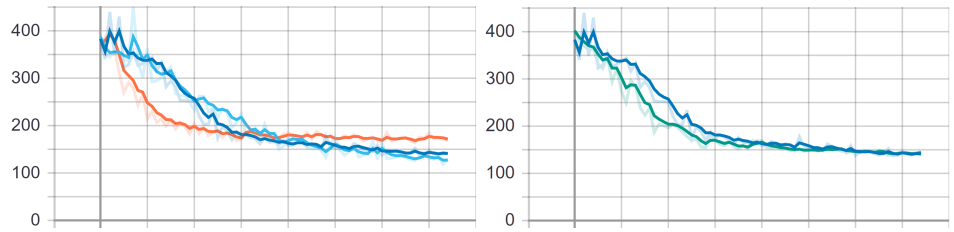
Figure 3: Generated MNIST digits.

As previously mentioned, some hyper-parameters for the DCGAN are perceived *a priori* as ideal based on suggestions made in the original paper [5]. Along with these suggestions, a guideline was also provided which contained claims, such as use Leaky ReLU activation in the discriminator and ReLU in the generator. As such, part of this project also intends to check whether these suggestions indeed improve the performance of our model. For that, we will start by assuming a so called base model that will serve as measure of comparison. This model contains the following parameters: *learning rate*= 1e-4, *batch size*= 64 and $\beta_1 = 0.9$.

Due to the computational cost of training with the images contained in the celebA dataset, most experiments were performed on a reduced size (32x32). These experiments are mostly related to small hyper-parameter changes that were claimed in previous research. Performance evaluation in GANs is not as simple as with other networks, here we don't really have an objective loss function and (manually) evaluating the generated result would ultimately be a ambiguous evaluation and prone to biases. Thus, in this project we make use of a quantitative measure confirmed to make a correct assessment of the model, the FID. With this measure, the lower the score the better.

Starting by experimenting with the batch size, we tested three values (32, 64 and 128) which were assumed to be optimal. Looking at Figure 4a we can observe how the final FID score after 75 epochs is quite similar between the three curves, although batch size 32 has a slightly higher value. The most significant difference between 64 and 128, however, is the training time. From all of them, using a batch size of 128 allowed for a faster training (performing at almost twice the speed) and a lower result.

Moving on to the learning rate, we compared the values 1e-4, 1e-3 and 2e-4. In the original paper, it is suggested that the second value would be too high, thus better results would be obtained with the third value. Figure 4b shows how the performance varies with these values. Attentive eyes might notice how only two curves are displayed, that is because with a learning rate of 1e-3 the gradients exploded right away thus there was no need to proceed with that specific experiment. Nevertheless, comparing the base model value of 1e-4 with the suggested value of 2e-4 we conclude that there is not any significant difference between the two, thus we pick 1e-4 to proceed with. Additionally, as the generator and discriminator are implemented separately one may consider applying different learning rates to both. That situation was also tested on this model, but no positive outcome came with it, neither in terms of lower FID score or training speed.



(a) (b)

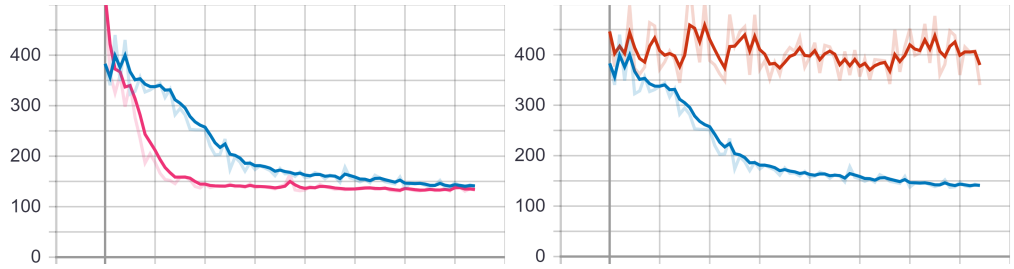
Figure 4: FID score comparison for (a) batch size (32 in orange, 64 in dark blue and 128 in light blue) and (b) learning rate (1e-4 in blue and 2e-4 in green). The y-axis represents the FID score, and the x-axis represents the epoch number up to 75 epochs.

The last parameter change suggestion from the paper refers to the β_1 value of the optimization algorithm *Adam*. Here we compared the 0.9 used in the base model to the suggested 0.5. The results were quite significant (Figure 5a) as using $\beta_1 = 0.5$ allowed for a more stabilized training, avoiding the oscillating present in the other curve. The FID score had an exponential decrease in the first 20 epochs reaching "convergence" around 130.

Tests on the network's architecture were also performed. In the guideline presented in [5] the authors claim that using ReLU in the generator (and Leaky ReLU in the discriminator) would maximize the performance. As we can observe for the red curve in Figure 5b), this was not confirmed. The explanation for this is that sparse gradients (ReLU) usually lead to problems in the stability of training, thus the presence of those oscillations.

Moreover, changing the number of layers did not provide better results, often leading to the network not running at all.

As an optimization procedure, we augmented the data, by randomly flipping 20% of the images horizontally or vertically, to check whether that would indeed improve the result. After comparison against a baseline model it was concluded that this random augmentation did not yield a significant improvement in the FID score.



(a) (b)

Figure 5: FID score comparison for (a) β_1 (0.9 in blue and 0.5 in red) and (b) Leaky ReLU vs ReLU (red curve corresponds to ReLU and blue to Leaky). The y-axis represents the FID score, and the x-axis represents the epoch number up to 75 epochs.

All in all, considering the tested parameters, the best model was found to be using $\beta_1 = 0.5$, batch size = 64, learning rate = 1e-4. We allowed this final model to run for 200 epochs and the resulting FID is 102, which is lower than any of the previous experiments. The generated images can be observed in Figure 6. The quality of the generated faces is good, however as it is down-scaled to 32x32 pixels the images lacks a bit of detail.

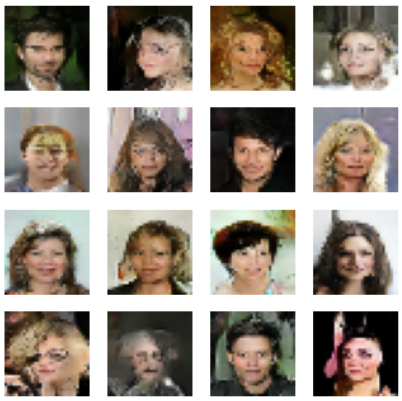


Figure 6: Faces generated by the optimal DCGAN model.

When attempting to run the same network on 64x64 there was problem in getting the network to learn how to produce realistic faces. It was then found that the hyper-parameter settings from the original paper was better suited for images with higher resolution.

The result of running the original settings on 64x64 can be seen in Figure 7. In this figure we can see some faces, such as the one generated in the top right, looks quite realistic. However, there are still cases when the network generates noisy images.

5 Conclusions

In this report, we re-implemented a DCGAN model, testing it in a celebrities dataset with the goal of generating, as realistic as possible, fake human faces. Due to the computational cost, we stuck with a 32x32 image size and compared an arbitrary base model with some suggestions provided in [5]. Our results showed that the best model can be obtained using a learning rate of 1e-4, batch size of 128 and β_1 of 0.5. We also found that the suggested usage of a ReLU activation in the generator (with Leaky ReLU in discriminator) affects the stability of the training, thus it is optimal to just use Leaky in both.

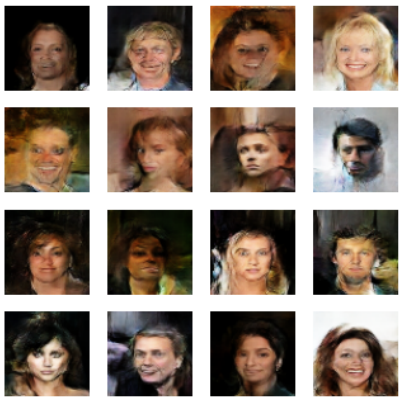


Figure 7: Faces generated by the base model on 64x64.

With this implementation we were able to obtain good results, but for an even greater performance one would need to implement some optimization methods. For instance, although we did implement a basic data augmentation technique which did not allowed for significantly better results, in [7] a data augmentation method for GANs was introduced and we believe that it would improve the performance of our model.

Further implementations of DCGAN were also discussed, for instance, vector arithmetic for visual concepts (as demonstrated in [5]) where we can subtract, for example, the face of a human with glasses to a face of the same man without glasses and then add the face a woman, which will result in the face of the woman with glasses.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[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, 2018.

[3] Huaibo Huang, Zhihang Li, Ran He, Zhenan Sun, and Tieniu Tan. Introvae: Introspective variational autoencoders for photographic image synthesis. *arXiv preprint arXiv:1807.06358*, 2018.

[4] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *ICCV*, pages 3730–3738. IEEE Computer Society, 2015.

[5] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016.

[6] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, Jan 2015.

[7] Ngoc-Trung Tran, Viet-Hung Tran, Ngoc-Bao Nguyen, Trung-Kien Nguyen, and Ngai-Man Cheung. Towards good practices for data augmentation in GAN training. *CoRR*, abs/2006.05338, 2020.