DeepFake Generation in the world of Deep Convolutional Generative Adversarial Networks (DCGAN)

Sharanya Venkat
Computer Science & Engineering
PES University
Bangalore, India
sharan.venkat@gmail.com

Richa
Computer Science & Engineering
PES University
Bangalore, India
richa5june99@gmail.com

Gaurang Rao
Computer Science & Engineering
PES University
Bangalore, India
gaurangpesu@gmail.com

***Abstract*—Deep learning, has been successfully employed in various fields that require complex problem solving, including big data analytics and computer vision. More recently, deep learning has introduced a set of very interesting algorithms, under the umbrella of Deep-Fake, where these algorithms can generate fake audio, images and videos, that are hard to distinguish from authentic ones. This paper presents experimentation on generation of images with faces, using Deep Convolutional Generative Adversarial Networks (DCGANs).**

I. Introduction

Learning reusable feature representations, from large unlabeled data sets has recently been an area of active research. In broad terms, Generative Adversarial Networks [1] (GANs), use the concept of two deep networks - a generator G and a discriminator D, competing against each other, where the generator tries to produce output such that the discriminator cannot distinguish it to be fake, and the discriminator aims to beat the generator, by classifying the generator's output as fake. A generative model G captures the data distribution, and a discriminative model D estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. Deep Convolutional GANs, take this a step further in the world of Machine Learning, by replacing the dense layers in GANs, with Convolutional layers. We introduce a class of CNNs called deep convolutional generative adversarial networks (DCGANs), that have certain architectural constraints, and demonstrate that they are a strong candidate for unsupervised learning.

Deep-fake algorithms (as implemented in [2], [3]) generally require a large amount of image and video data to train models to create realistic looking images and videos. As public figures, such as celebrities and politicians are expected to have a large number of videos and images available online, they are the initial targets of deep-fakes.

In our implementation, we train our model on two popular and easily available data sets - CelebA and Celebrity-100k, both of which are repositories containing 100k-200k images of well known celebrities. We present a variety of experiments on the above data sets, by tuning different hyper-parameters. In **Creating the Deep-fake Generation Network**, we present the principles of deep-fake algorithms and deep-fake generation and how deep learning has been used to enable such disruptive technologies. **Experiments** presents the range of experiments carried out. We discuss challenges, research trends and as well as our future goals regarding the project in **Challenges Faced**.

II. Creating the Deep-fake Generation Network

A. Model Architecture

Deep learning is well known for its capability of representing complex and high-dimensional data. One variant of the deep networks with that capability is Generative Adversarial Networks. GANs learn a probability distribution of a data set by pitting two neural networks against each other. One neural network, called the Generator, generates new data instances, while the other, the Discriminator, evaluates them for authenticity; i.e, the discriminator decides whether each instance of data that it reviews belongs to the actual training data set or not. Meanwhile, the generator is creating new, synthetic/fake images that it passes to the discriminator. It does so in the hopes that they too will be deemed authentic, even though they are fake. The fake image is generated from a 200-dimensional noise (uniform distribution between -1.0 to 1.0) using the inverse of convolution, called transposed convolution. The goal of the generator is to generate passable images i.e to lie without being caught. The goal of the discriminator is to identify images coming from the generator as fake.

Here are the steps a GAN takes:

- The generator takes in random noise and returns an image.
- This generated image is fed into the discriminator alongside a stream of images taken from the actual, ground-truth data set.
- The discriminator takes in both real and fake images and returns probabilities, a number between 0 and 1, with 1 representing a prediction of authenticity and 0 representing fake.

A double feedback loop is created as follows:

- The discriminator is in a feedback loop with the ground truth of the images, which we know.
- The generator is in a feedback loop with the discriminator.

To begin with, we normalise the images by loading them using PIL - Python Imaging Library. While loading the images we crop all images to center them around the face, and resize them to an image with (64, 64, 3) dimensions. We pass a uniformly distributed noise to the Generator. The generator converts this noise into an image of (64, 64, 3) size. We use transpose convolution in this process. Batch normalization layers are used after the transpose convolution layer instead of the last layer. We use variations of ReLU and leaky ReLU activations after batch normalization layers. We pass our final image through a tanh activation to squash the pixel range between (-1, 1). We pass a tensor of shape (64, 64, 3) to the discriminator. Discriminator gives a an output telling whether this image is real or fake. We pass the final output through sigmoid activation. Which squashes the output between (0, 1). If the output is close to 1, the discriminator identifies the image as a real image and if the output is close to 0, the image is identified as a fake image.

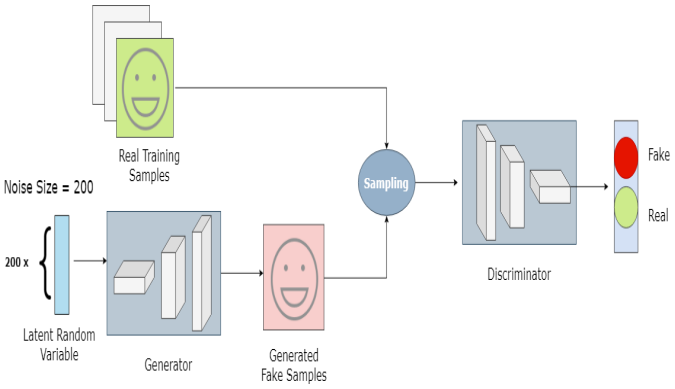

Fig. 1. Architecture of a GAN.

B. *Details of Adversarial Training of DCGANs*

The convolutional networks in our model are trained to beat their opponent, therefore, it is important for both the networks to realise the output of their adversary. The generator starts by taking noise as input from a uniform distribution, and runs it through its network, in attempt to create an image that can fool the discriminator. The discriminator trains, by taking the generators new image, along with training data received from a data set, trying to learn what a real image looks like, and aiming to realise the generators output to be fake. Based on the images received from the training data, the weights within the discriminator network are adjusted. The generators loss function is in turn, made to depend on the discriminators output, when it is provided with the image the generator created, as input. This way, the generator gets an idea of how the output of the discriminator looks like when it provides its freshly created image as input, and on the other side, the discriminator is learning, by understanding training images, and tuning its weights to classify such an image as real. A successful training of the networks would lead to the generator creating such a realistic-looking image, that the discriminator is unable to realise that the image is in-fact fake.

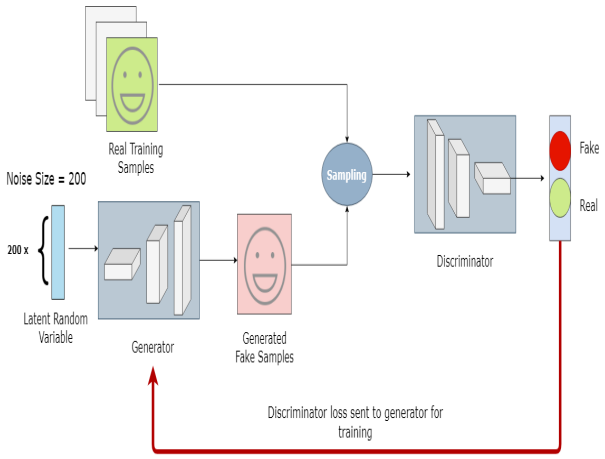

Fig. 2. Training Process for Generator

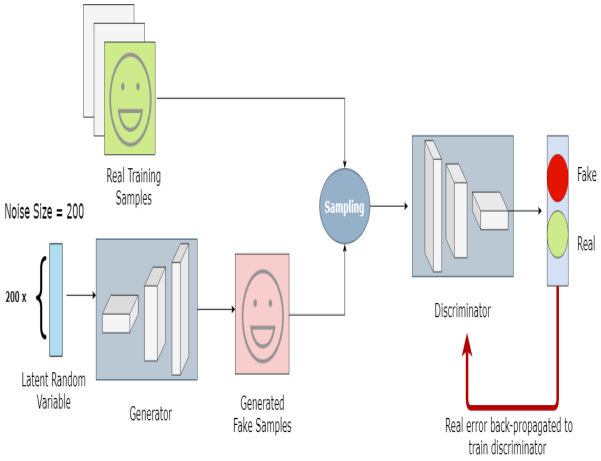

Fig. 3. Training Process for Discriminator

C. *Formulation of GAN*

As we know the discriminator is a binary classifier so when we feed the real data , the model should produce high probability for the real data and low probability for fake data or the generator's output. Inspired by game theory the generator and discriminator are pit against each other during the training. The generator G gets stronger and stronger at generating the real type of results and the discriminator D also gets stronger and stronger at identifying which one is real, which one is fake. Hence it takes the form of adversarial training. In order to understand the algorithm behind training GANs let us first define a few variables and functions that would be commonly addressed and used further.

Let,

$\mathbf{z} \longrightarrow$ Noise Vector

$\mathbf{G(z)} \longrightarrow$ Generators Output or $\mathbf{X_{fake}}$

$\mathbf{x} \longrightarrow$ Real Training Samples or $\mathbf{X_{real}}$

$\mathbf{D(x)} \longrightarrow$ Discriminator's output for $\mathbf{X_{real}}$

Let $P(y|x_{\text{real}})$ be the output of the discriminator for X_{real} that lies between {0,1}

$\mathbf{D(G(z))} \longrightarrow$ Discriminator's output for X_{fake}

Let $P(y|x_{\text{fake}})$ be the output of the discriminator for X_{fake} that lies between {0,1}

At Discriminator D the following needs to be done.

D(x) $\longrightarrow$ should be **maximised** and

D(G(z)) $\longrightarrow$ should be **minimised**

At Generator G the following needs to be done.

D(G(z)) $\longrightarrow$ should be **maximised**

As you can see the **D(x), D(G(z))** give a score between 0 and 1. We want to build a model (discriminator) that maximizes the real data while minimizing the fake data and **G(z)** gives the same shape as the real input (ex: if image of 10*10 is the real input then **G(z)** produces the same shape but, of course, it's noisy). We also want to build a model(generator) that maximizes the fake data.

At Discriminator D we see that,

$$Dloss_{\text{real}} = log(D(x))$$

$$Dloss_{\text{fake}} = log(1 - D(G(z)))$$

$$Dloss = Dloss_{\text{real}} + Dloss_{\text{fake}}$$

The total cost is,

$$\frac{1}{m}\sum_{i=1}^{m} log(D(x^i)) + log(1 - D(G(z^i)))$$

At Generator G we see that,

$$Gloss = log(1 - D(G(z)))$$

or

$$Gloss = -log(D(G(z))$$

The total cost is,

$$\frac{1}{m}\sum_{i=1}^{m} log(1 - D(G(z^i)))$$

or

$$\frac{1}{m}\sum_{i=1}^{m} -log(D(G(z^i)))$$

The discriminator network runs twice (one for real, one for fake) before it calculates the final loss while generator runs only once. Once we got these two losses, we calculate the gradients with respect to their parameters and back propagate through their networks independently. (as discussed in detail in [1]) We can view this as D and G playing the following two-player mini-max game with value function $V(D, G)$. This value function in terms of expectation is defined as,

$$\min_G \max_D V(D,G) = \mathbb{E}_{\text{z} \backsim \text{pdata(x)}} log(D(x)) + \mathbb{E}_{\text{z} \backsim \text{p}_{\text{z(z)}}} log(1 - D(G(z)))$$

$$\max_D V(D) = \mathbb{E}_{\text{z} \backsim \text{pdata(x)}} log(D(x)) + \mathbb{E}_{\text{z} \backsim \text{p}_{\text{z(z)}}} log(1 - D(G(z)))$$

Here the first term in the RHS of the equation helps in recognising real images better whereas the second term in the RHS of the equation helps in recognising generated fake images better.

$$\min_G V(G) = \mathbb{E}_{\text{z} \backsim \text{p}_{\text{z(z)}}} log(1 - D(G(z)))$$

The term in the RHS of the equation helps in optimising G that can fool the discriminator the most.

III. Experiments

We run our model on two different data sets which are celebrity-100k and celebA, both containing 100k-200k high quality images of well known celebrities. The parameters chosen to tune include learning rate, the size of noise, leak parameter α for LeakyRelu, label smoothing factor and β value for Adam optimiser. The following subsections elaborate the details of the experiment.

A. *Celebrity-100k Dataset*

We carried out two variations of experiments on the Celebrity-100k data set containing 100k high quality images of well known celebrities.

The first experiment ran with the following parameters : batch size as 128, β = 0.5, label smoothening factor= 0.9, noise size = 200, with the learning rate as 0.0001 and input shape as (64,64,3). ReLU activation was used after batch normalization layers and the model was trained for 30 epochs given the hardware constraints and size of the data set used. For the above implementation, the generator failed to produce unique outputs. We believe that the reason for this is that the generator caused a model collapse failure. As we see in Figure 4, the generator seems to be reproducing the same set of features repeatedly.The reason for this needs to be further investigated. This challenge is further is addressed in section **Challenges Faced** in detail.

Figure 5 shows the plot of the loss output by the discriminator against the loss of the generator.

On a basis of trial and error we landed at an ideal learning rate which was 0.0002 instead of the initial value 0.001. We also extended our experiment and replaced ReLU with leaky ReLU and found a significant change in the output. The model performed very well generating a variety of unique deepfakes. Figure 6 shows the images generated by the model The plot in Figure 7 depicts the loss generated by discriminator plotted against the loss of the generator. As we can see, convergence is reached when discriminator's loss is maximised and generator's loss is minimised i.e, the discriminator is no longer able to distinguish between real and fake images.

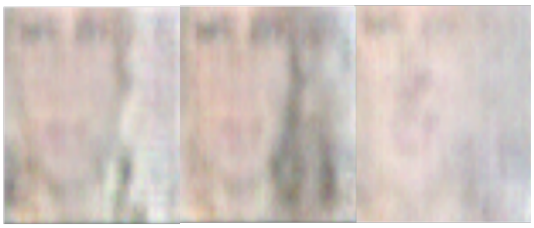

Fig. 4. Images generated by the model

Fig. 5. Plot of the Loss generated by the model when trained using ReLU

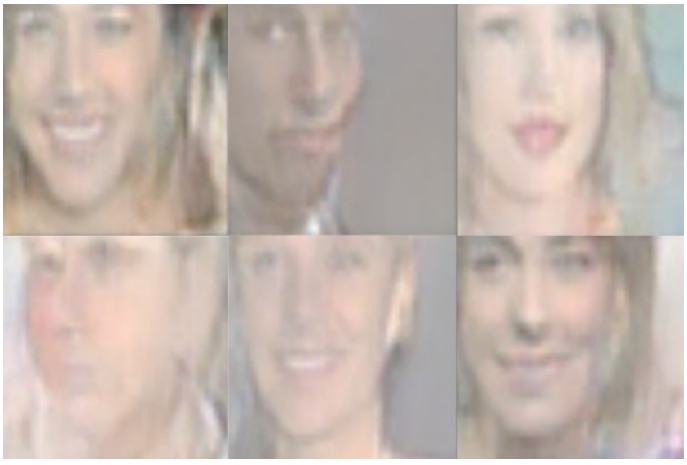

Fig. 6. Images generated by the model

Fig. 7. Plot of the Loss generated by the model when trained using leaky-ReLU on Celebrity-100k data set

B. CelebA data set

We show experiment results for another popular data set called CelebA containing 200k high quality images of well known celebrities. The experiment ran with the following parameters : batch size as 128, β = 0.5, label smoothening factor= 0.9, noise size = 200,with the learning rate as 0.0001 and input shape as (64,64,3). Leaky-ReLU activation was used after batch normalization layers and the model was trained for 10 epochs given the hardware constraints and size of the dataset used. Figure 8 shows the images generated by the network after training.

Fig. 8. Images generated by the model

The plot in Figure 9 depicts the loss generated by discriminator plotted against the loss of the generator

IV. Challenges Faced

GANs have a number of common failure modes. All of these common problems are areas of active research. While none of these problems have been completely solved, there has been progress in the work carried out by various researchers

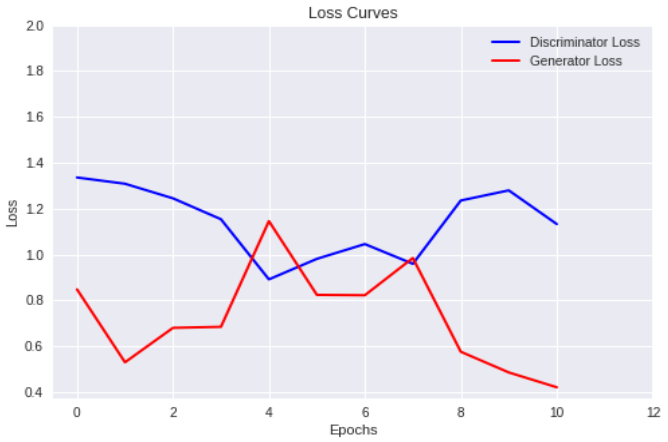

Fig. 9. Plot of the Loss generated by the model when trained using leaky-ReLU on CelebA dataset

around the globe regarding the same. We try and address some common problems faced in training and possible measures to overcome them

- **Vanishing Gradients:**
  Research has suggested that if your discriminator is too good, then generator training can fail due to vanishing gradients. In effect, an optimal discriminator doesn't provide enough information for the generator to make progress.
- **Model Collapse:**
  Generally, you want your GAN to produce a wide variety of outputs. You want, for example, a different face for every random input to your face generator.
  However, if a generator produces an especially plausible output, the generator may learn to produce only that output. In fact, the generator is always trying to find the one output that seems most plausible to the discriminator. If the generator starts producing the same output (or a small set of outputs) over and over again, the discriminator's best strategy is to learn to always reject that output. But if the next generation of discriminator gets stuck in a local minimum and doesn't find the best strategy, then it's too easy for the next generator iteration to find the most plausible output for the current discriminator.
  Each iteration of generator over-optimizes for a particular discriminator, and the discriminator never manages to learn its way out of the trap. As a result the generators rotate through a small set of output types. This form of GAN failure is called model collapse.
  In the first experiment carried out we believe we ran into model collapse.
- **Failure to Converge**
  GANs usually fail to converge often. Some measures to overcome this include adding noise to discriminator input.

V. Conclusions and Future Research Directions

With our above implementation and experimentation we can conclude that it's definitely possible to generate lifelike looking faces with generative adversarial networks. It is viable even with very limited resources and hence it would be possible to render better and higher resolution samples in bigger and more advanced research labs. We can modify our network and generate more realistic high-resolution images by making our network deeper, but it comes at a cost of resources and time taken to train.

Our proposed future directions of work include being able to extend the current implementation to generate better and higher resolution images using Super Resolution GANs(SR-GANs) which is an upcoming architecture and a hot spot for research and experimentation. We would also like to extend it generating deep-fake videos. Another research approach is to improve the performance of detection methods for detecting deep-fake images, videos and audios.

VI. Acknowledgements

We would like to thank the Department of Computer Science and Engineering, PES University, for giving us this opportunity to do a project of such magnitude for the course **Topics in Deep Learning**.

We would like to thank Prof. Srikanth H R and Prof. Srinivas Kathargуppe for their constant support and guidance throughout the semester. We are grateful to them for sharing their knowledge with us and encouraging us to experiment with new and innovative ideas which has helped us in improving our skills and interests in deep learning.

We would like to thank the chairperson, Dr. Shylaja S Sharath for giving us this opportunity and for her continuous support.

References

[1] Ian J. Goodfellow, Jean Pouget-Abadie ,Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair†, Aaron Courville, Yoshua Bengio *Generative Adversarial Nets*. 2014

[2] Thanh Thi Nguyen, Cuong M. Nguyen, et.al. *Deep Learning for Deepfakes Creation and Detection*. Deakin University, Australia, 2019.

[3] Alec Radford, Luke Metz, Soumith Chintala. *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*. 2016.