Project Presentation

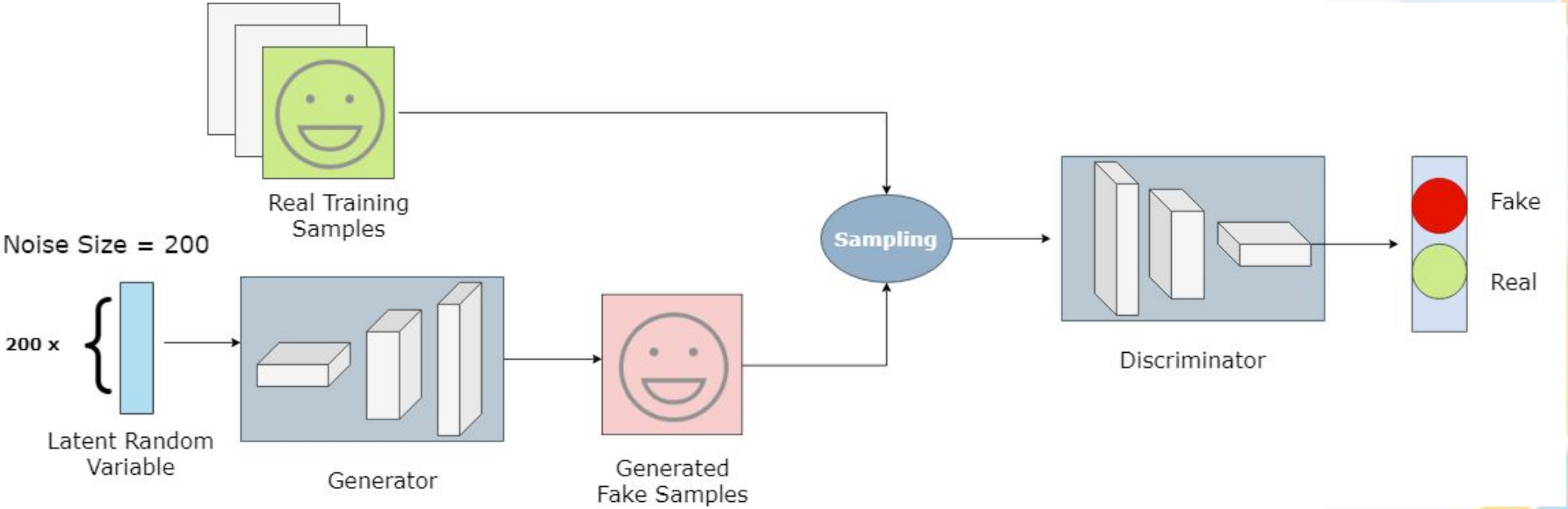
(Final - ESA)

Project Title	:	Deep-Fake Generation in the world of Deep Convolutional Generative Adversarial Networks (DCGAN)	
Project Team	:	Sharanya Venkat	PES1201700218
		Richa	PES1201700688
		Gaurang Rao	PES1201701103

Project Abstract and Scope

- We strongly believe that there is large scope in the field of Deep-Fake with the world in the 4th industrial revolution.
- Websites generating fake faces have been trending on the internet with millions of views.
- We were eager to explore the topic and understand the math behind it to implement a deep-fake algorithm on our own.
- As compared to solutions that exist, our solution uses the concept of DCGANs which differs from simple GANs.
- Our solution is unique because we introduce the concept of convolutional networks to simple GANs.
- It essentially looks for spatial correlation which is more fitting for the concept of fake face generation.

Solution Architecture



The components that go into the solution for generating fake faces are:

1. Training data → consists of a large dataset of images of faces
2. Generator → generates fake images
3. Discriminator → takes the training data and the fake images,and classifies them as fake or real

Adversarial Training Algorithm

Algorithm:

- The generator takes in random noise from a uniform distribution 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.

GAN Formulation

$\mathbf{z} \longrightarrow$ 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 $\mathrm{X_{real}}$ that lies between {0,1}

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

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

At Discriminator D the following needs to be done.

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

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

At Generator G the following needs to be done.

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

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,

$$\min_G \max_D V(D,G) = E_{\mathrm{z\ pdata(x)}} log(D(x)) + E_{\mathrm{z\ p_z(z)}} log(1 - D(G(z)))$$

Experimentation

We carried out the experiment on two datasets to get a better insight.

1. Celebrity-100k Dataset:

Test 1:

β (for ADAM optimizer) = 0.5
α = 0 (ReLU)
noise_size = 200
learning_rate = 0.0001

Test 2:

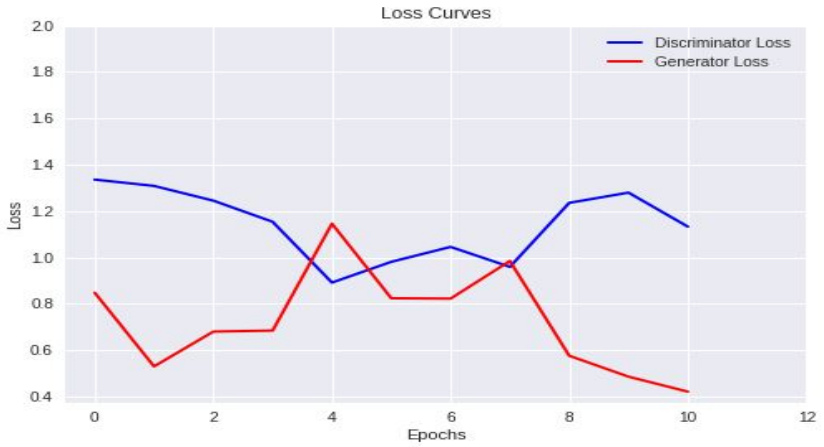
β (for ADAM optimizer) = 0.5
α (for Leaky ReLU) = 0.2
noise_size = 200
learning_rate = 0.0002

2. CelebA dataset:

Test 1 (similar to Test 2):

β (for ADAM optimizer) = 0.5
α (for Leaky ReLU) = 0.2
noise_size = 200
learning_rate = 0.0002

Analysing the Solution



- A successful training of the networks would lead to the generator creating a realistic-looking image, which would deceive the discriminator into realising the image to be real.
- This means that on convergence, the loss of the discriminator is maximised, and the loss of generator is minimised, as precisely depicted by the graphs above which are a result of our experiments.

Constraints, Assumptions & Dependencies

- DCGANs involve training deep convolutional networks, which adds certain hardware constraints, making it strenuous to run on a local system. (All our experiments were run on cloud servers such as Google Colab and Kaggle.)

- Training DCGANs demanded a lot of time, (considering the size of our dataset), and we had to use a GPU as an accelerator, on Kaggle. Even then, we were forced to run it for a limited number of epochs, due to time constraints, thus assuming that the output of our program was adequate.

- Since we worked with TensorFlow version 1.15.0, we were required to downgrade from the default TensorFlow version, along with the TensorFlow GPU versions on these cloud services, to 1.15.0.

Future Work Plan

- 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 (SRGANs), 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.

References

- Ian J. Goodfellow, Jean Pouget-Abadie, et. al
  Generative Adversarial Nets
  Universite de Montreal, Montreal, 2014.
  https://arxiv.org/pdf/1406.2661.pdf

- Thanh Thi Nguyen, Cuong M. Nguyen, et. al
  Deep Learning for Deepfakes Creation and Detection
  Deakin University, Australia, 2019.
  https://www.researchgate.net/publication/336055871_Deep_Learning_for_Deepfakes_Creation_and_Detection

- Alec Radford, Luke Metz, Soumith Chintala.
  Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks
  2016.
  https://arxiv.org/abs/1511.06434

- https://developers.google.com/machine-learning/gan/gan_structure

Please find our citations in the report for the above papers.

Thank You