Face Generator Using DCGANs

A Project Report

Submitted to the National Institute of Technology Karnataka -Surathkal

in partial fulfillment of requirements for the awardee of Semister-1

Master of Technology

in

Signal Processing and Machine Learning

by

Posibabu Sankarapu & Shiva Sai Samboju

(202SP020 & 202SP023)

DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING

NIT K Surathkal

Mangalore

January 2021

DEPT. OF ELECTRONICS & COMMUNICATION ENGINEERING

National Institute of Technology Surathkal -Mangalore

2020 - 21

CERTIFICATE

This is to certify that the Project report entitled **Face Generator Using DCGANs** submitted by **Posibabu Sankarapu & Shiva Sai Samboju** ((202SP020 & 202SP023)), to the NITK Surathkal in partial fulfillment of the M.Tech. degree in Signal Processing and Machine Learning is a bonafide record of the project work carried out by him/her under our guidance and supervision. This report in any form has not been submitted to any other University or Institute for any purpose.

Prof.A.V.Narasimhadhan

Professor

Dept.of ECE

NIT K Surathkal

Mangalore

DECLARATION

We Posibabu Sankarapu & Shiva Sai Samboju hereby declare that the project report **Face Generator Using DCGANs** , submitted for partial fulfillment of the requirements for the evaluation of Master of Technology to the National institute of technology Surathkal, Mangalore is a bonafide work done by me under supervision of Prof.Shyam Lal

This submission represents my ideas in my own words and where ideas or words of others have been included, I have adequately and accurately cited and referenced the original sources.

I also declare that I have adhered to ethics of academic honesty and integrity and have not misrepresented or fabricated any data or idea or fact or source in my submission.

Posibabu Sankarapu & Shiva
Sai Samboju

Mangalore
24-01-2021

Abstract

In recent years, supervised learning with convolutional networks (CNNs) has seen huge adoption in computer vision applications. Comparatively, unsupervised learning with CNNs has received less attention. In this project we hope to help bridge the gap between the success of CNNs for supervised learning and unsupervised learning. 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. Training on various image datasets, we show convincing evidence that our deep convolutional adversarial pair learns a hierarchy of representations from object parts to scenes in both the generator and discriminator.

Key Words: Generator; Discriminator; data sets,Loss,Training

JIM

Contents

Chapter 1

Introduction

DCGAN is an extension of the pre-existing GAN network, but in DCGAN, convolutional networks and transposed convolution is used for upsampling. In this there two distinct models, a generator and a discriminator. The job of the generator is to produce random noise and map the noise to the images in the dataset. The job of the discriminator is to look at an image and output whether or not it is a real training image or a fake image from the generator. During training both generator and discriminator becomes better in their respective tasks and equilibrium is reached when discriminator is left to guess whether the image is from real dataset or fake with 50 % confidence
In practice GAN's are very hard to train mainly because of the following reasons :

1. The Nash Equilibrium is very hard to reach using Gradient Descent based methods used for training

2. GAN's tend to collapse to a single mode during training thus generating a single sample in majority.

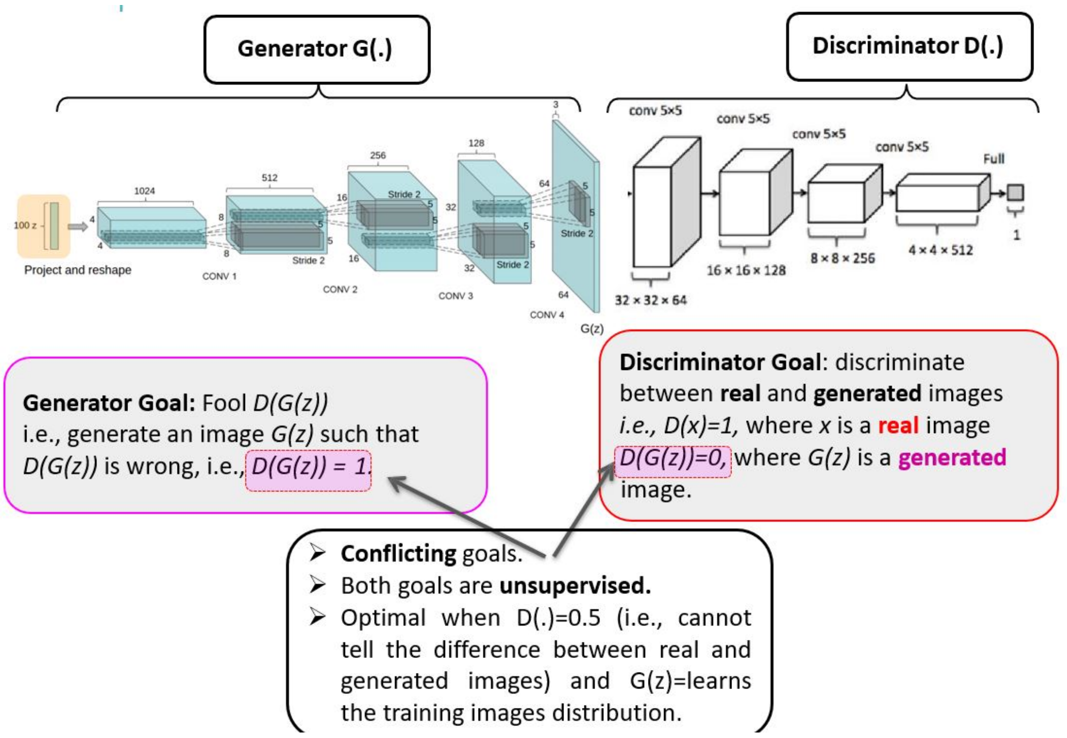
3. GAN training is unstable and requires a lot of model exploration and careful selection of hyperparameters.

DCGANS try to solve some of the above problems by using ConvNets in the Generator and Discriminator instead of simple MLP's. The authors in the paper also propose some architectual contraints on the ConvNets that helps stabilize GAN training[1]. Some of these constraints are:

1. Using only convolutional layers in the Generator and the Discriminator by increasing the stride, therfore discouraging the use of MaxPooling and Dense layers in the ConvNet architecture

2. Using BatchNormalization in both the Generator and the Discriminator during training. However, during training I observed that using BatchNorm in all the layers causes the training to oscillate and destabilize.

3. The generator uses tanh activation in the final layer while the discriminator uses leaky-relu activation in all the layers.



Generator Goal: Fool *D(G(z))*
i.e., generate an image *G(z)* such that *D(G(z))* is wrong, i.e., *D(G(z)) = 1.*

Discriminator Goal: discriminate between **real** and **generated** images *i.e., D(x)=1,* where *x* is a **real** image *D(G(z))=0,* where *G(z)* is a **generated** image.

- **Conflicting** goals.
- Both goals are **unsupervised.**
- Optimal when D(.)=0.5 (i.e., cannot tell the difference between real and generated images) and G(z)=learns the training images distribution.

(a) DCGAN's Architecture

Figure 1.1: Deep convolutional Generative Adversarial network

Chapter 2

Literature Review

Generative adversarial networks (GANs) introduced by Ian Goodfellow in 2014. GANs is completely new way of teaching computers how they do complex tasks through a generative process. Yann LeCun, one of the most prominent person in deep learning world, mention in his talk that “Adversarial training is really cool idea, it like coolest idea in machine learning in the last 20 years[2].” GANs have two components.

1. A Generator(An artist) neural network.
2. A Discriminator(An art critic) neural network.

Generator(An artist) generates an an image. The Generator does not know anything about the real images and learns by interacting with the Discriminator. Discriminator(An art critic) determines whether an object is “real” and “fake” (usually represented by a value close to 1 or 0)

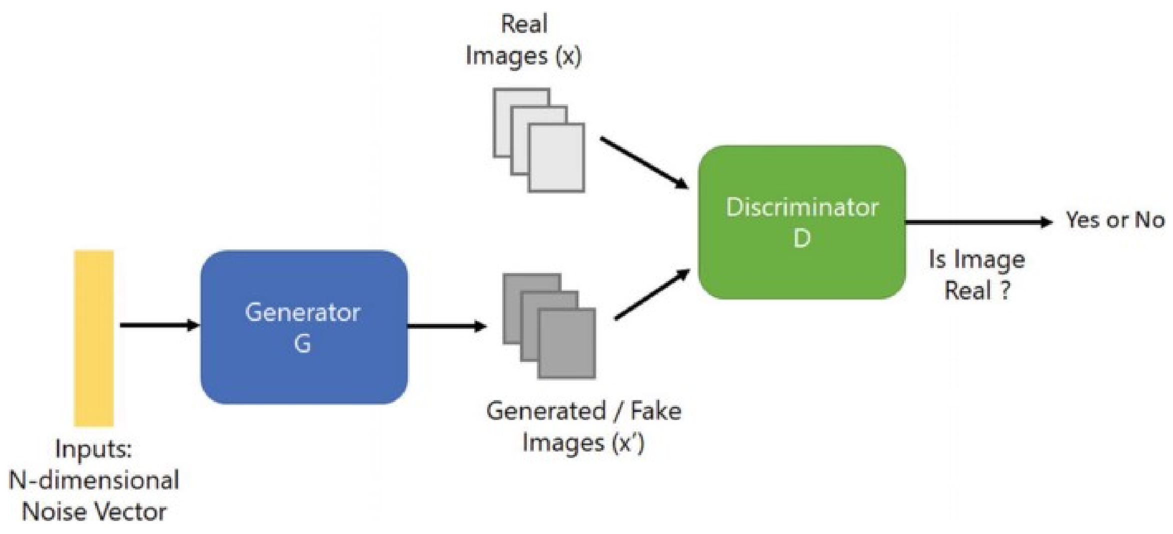


Figure 2.1: High level Architecture

2.1 Working of DCGANs

1. The Discriminator is a classifier that determines whether the given image is “real” and “fake”.
2. The Generator takes a randomly generated noise vector as input data and feedback from the Discriminator and generates new images that are as close to real images as possible[3].
3. The Discriminator uses the output of the Generator as training data.
4. The Generator gets feedback from the Discriminator.
5. These two models “battle” to each other. Each models becomes stronger in the process.
6. The Generator keeps creating new images and refining its process until the Discriminator can no longer tell the difference between the generated images and the real training images.

Architectural Constraints:

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator.
- Use LeakyReLU activation in the discriminator.

Original DCGAN architecture(Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks) have four convolutional layers for the Discriminator and four “four fractionally-strided convolutions” layers for the Generator.

Discriminator Network: The Discriminator is the “art critic” who tries to distinguish between “real” and “fake” images. This is a convolutional neural network for image classification.
The Discriminator is a 4 layers strided convolutions with batch normalization (except its input layer) and leaky ReLU activations. Leaky ReLU help the gradients flow easier through the architecture[4].

ReLU v/s Leaky ReLU

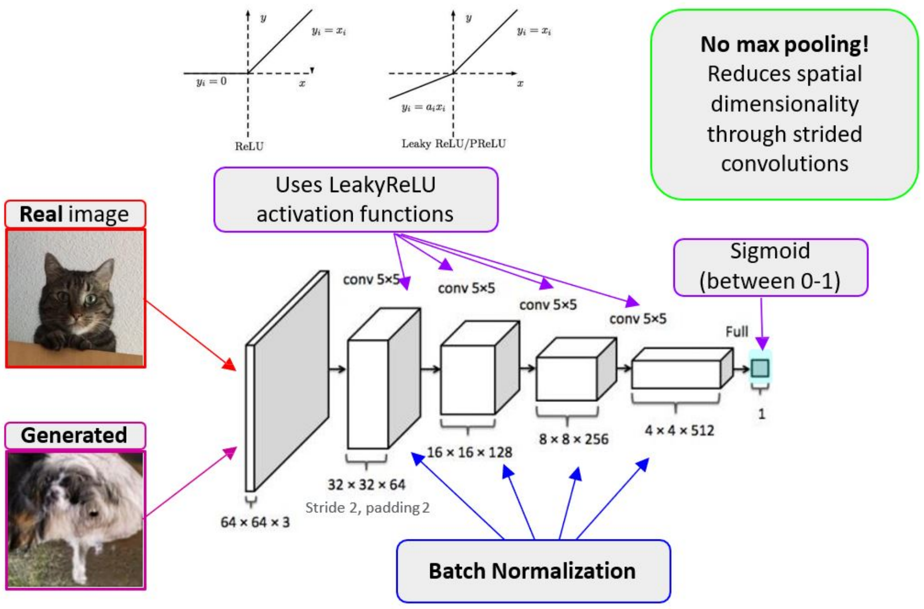
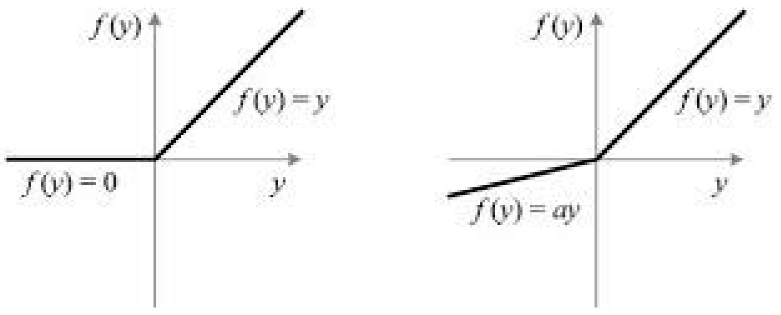


Figure 2.2: Discriminator network in DCGANs

- The ReLU activation function will just take the maximum between the input value and zero. If we use the ReLU activation function, sometimes the network gets stuck in a popular state called the dying state, and that's because the network produces nothing but zeros for all the outputs.

- Leaky ReLU prevent this dying state by allowing some negative values to pass through. The whole idea behind making the Generator work is to receive gradient values from the Discriminator, and if the network is stuck in a dying state situation, the learning process won't happen.

- The output of the Leaky ReLU activation function will be positive if the input is positive, and it will be a controlled negative value if the input is negative. Negative value is control by a parameter called alpha, which will introduce tolerance of the network by allowing some negative values to pass[5].



(a)

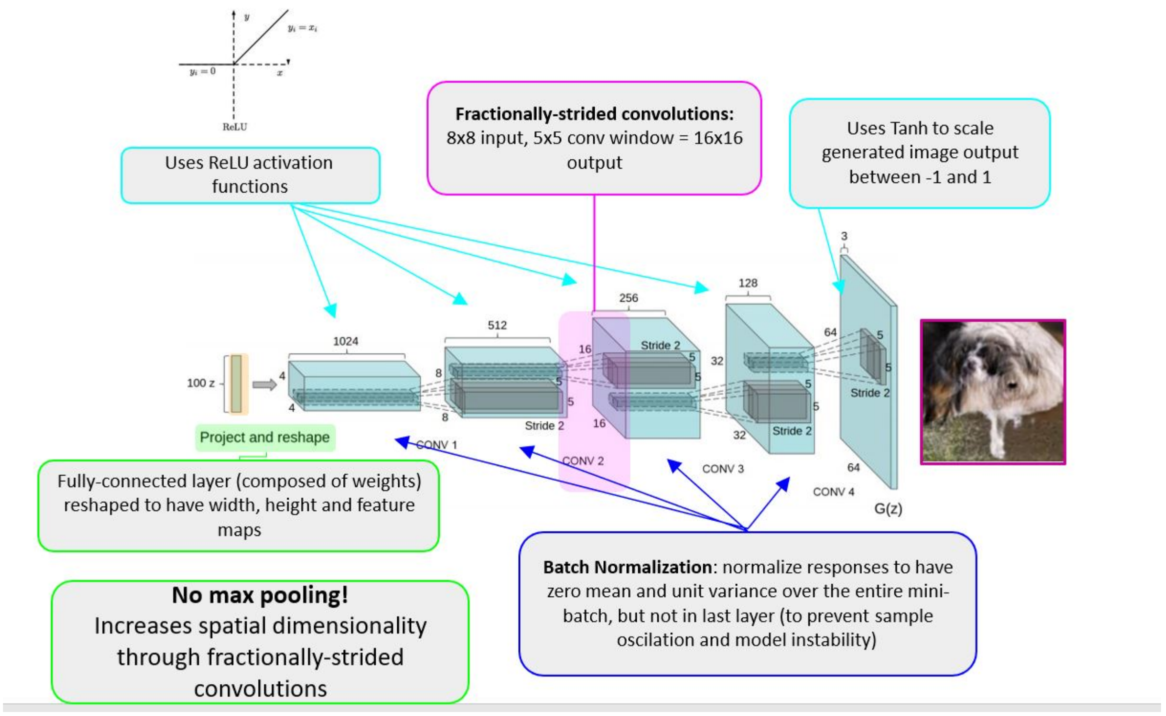
Figure 2.3: Relu vs Leaky Relu

Discriminator needs to output probabilities. For that, we use the Logistic Sigmoid activation function on the final logits.

Generator Network:

The Generator is the “An artist” who tries to create an images that looks as “real” as possible, to fool the Discriminator.

The Generator is a 4 layers fractional-strided convolutions with batch normalization (except its input layer) and use Hyperbolic Tangent (tanh) activation in the final output layer and Leaky ReLU in rest of the layers.



ReLU
Uses ReLU activation functions
Fractionally-strided convolutions: 8x8 input, 5x5 conv window = 16x16 output
Uses Tanh to scale generated image output between -1 and 1
100 z
Project and reshape
1024
512
256
128
3
Stride 2
CONV 1
CONV 2
CONV 3
CONV 4
G(z)
Fully-connected layer (composed of weights) reshaped to have width, height and feature maps
No max pooling! Increases spatial dimensionality through fractionally-strided convolutions
Batch Normalization: normalize responses to have zero mean and unit variance over the entire mini-batch, but not in last layer (to prevent sample oscilation and model instability)

Figure 2.4: High level Architecture

Loss-Generator and Discriminator losses:

The Generator is essentially trying to generate images that the discriminator would approve as real images. Hence, all the generated images must be predicted and as “1” (real) and must be penalized for failing to do so.

Hence, we train the generator to predict “1” as the output at the discriminator.

Contrary to the generator, the discriminator wants itself to predict generated outputs as fake, and at the same time, it must predict any real image as real. Hence, the discriminator trains on a combination of these two losses.

Optimization-Adam Optimizer:

We use the Adam optimizer to minimize the loss.

- Adam is different to classical stochastic gradient descent.
- Stochastic gradient descent maintains a constant learning rate (termed alpha) for all weight updates and the learning rate does not change during training.
- The Adam optimizer represents an alternative to the classical gradient descent method or stochastic gradient descent method.
- Adam adds Root Mean Square Deviation (RMSprop) to the process by applying per parameter learning weights. It analyzes how fast the means of the weights are changing and adapts the learning weights.

2.1.1 Implementation Details

Training of DCGANs:

The following steps are repeated in training

- First the Generator creates some new examples.
- The Discriminator is trained using real data and generated data.
- After the Discriminator has been trained, both models are trained together.
- The Discriminator's weights are frozen, but its gradients are used in the Generator model so that the Generator can update it's weights.

Intially we implemented DCGANs t DCGAN using TensorFlow and observed the results for two well-known datasets:

- MNIST handwritten digits dataset and
- CIFAR-10 image recognition dataset

After training, we obtained the following results,

On MNIST: for different epochs

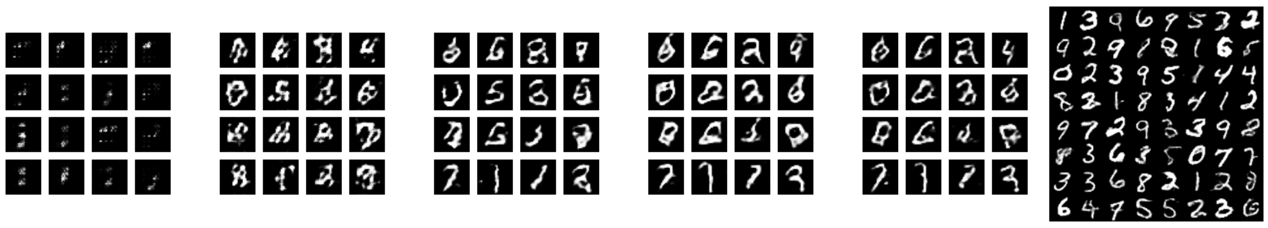


Figure 2.5: MNIST training with different epochs and batch sizes

On CIFAR-10:

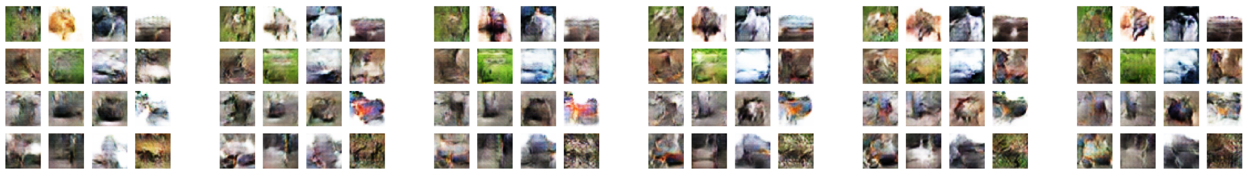


Figure 2.6: CIFAR-10 training with different epochs and batch sizes

Now, We implemented this architecture on CeleBA data set which is our original data set for this face generator

Data set and environment used:

Data set will be obtained from Kaggle, subset of celebrity dataset called celebA. It contains 100k images of cropped faces. The model will be trained by varying the following Hyperparameters:

- EPOCH
- Batch Rate
- Learning Rate

Goal:

Train to Achieve d.loss closer to 0.5 and g.loss closer to 1.0 which means discriminator does not know how to distinguish between real inputs and fake ones, and generator produces images so good that discriminator considers them as reals.

Chapter 3

Results and future work

We were able to train the model to generate near like images with D.loss=0.621 G.loss=0.756 for a batch of size 32.

Better results can be obtained by developing the model to learn images with annotations like age, Gender Facial features and accessories

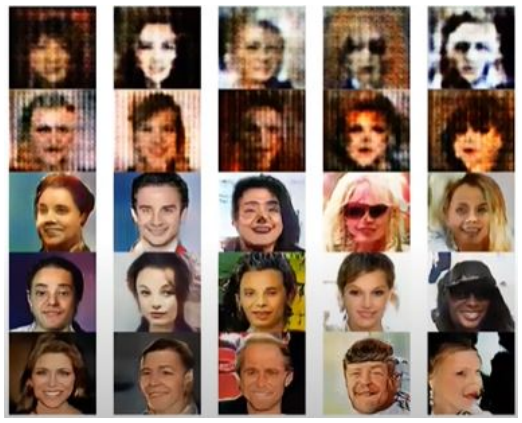


Figure 3.1: Batch size of 64 and Epoch 5 ,10 20 40

The best results were obtained from 3rd model trained as we see very fine quality images and desired losses.

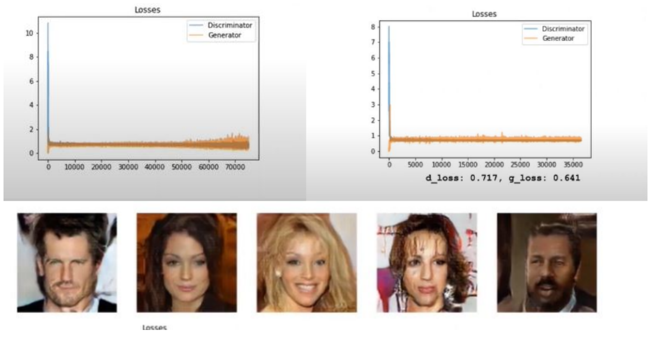


Figure 3.2: Batch size of 32 and Epoch 40 and Losses

In the Project of the analysis and experimentation, it was shown that many different models with different parameters give rise to different results, which in turn gave different losses, different times to run the codes and different metrics. As the exploration of different models and parameters were being done, there arose different questions, one included the validity metric(latent space) for real human datasets and generated faces.

As a conclusion note,DCGANs have shown tremendous potential in recent years and have been applied in various scenarios, ranging from image synthesis to enhancing the quality of images (superresolution), image-to-image translations, text-to-image generation, and more. In addition, GANs are the building blocks for advancements in the use of AI for art, music, and creativity (e.g., music generation, music accompaniment, poetry generation, etc.).

References

[1] Alec Radford , Luke Metz and Soumith Chintala *”Unsupervised representation learning with deep convolutional generative”* ,

[2] adversarial networks *ICLR 2016, arXiv preprint arXiv:1511.06434v2.*, Dept. Comp. Sci., 2006.

[3] Ian J. Goodfellow, Jean Pouget-Abadie†, Mehdi Mirza, Bing Xu,Sherjil Ozair‡, Aaron Courville, Yoshua Bengio, David Warde-Farley *Generative Adversarial Nets,*, Universit e de Montr ealMontr eal, QC H3C 3J7

[4] H. H. Barret, *Conditional generative adversarial nets for convolutional face generation*,

[5] *https://arxiv.org/abs/1511.06434* ,

[6] ” *https://towardsdatascience.com/dcgans-deep-convolutional-generative-adversarial-networks-c7f392c2c8f8*

[7] M. Turk and A. Pentland, *Eigenfaces for Recognition.*, J. Cognitive Neuroscience, vol. 3, no. 1, pp. 71-86, 1991.

[8] *https://pytorch.org/tutorials/beginner/dcgan$_f$aces$_t$utorial.html*

.