RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

RGUKT-Nuzvid, Eluru Dist – 521202

IMAGE GENERATION USING GENERATIVE ADVERSARIAL NETWORKS (GANs)

Report submitted to

Rajiv Gandhi University of Knowledge Technologies,

Nuzvid for the fulfillment of Mini project

of

Bachelor of Technology in

Computer Science and Engineering

by

N180114 (K. Sindhuja)

N180117(A. Pushpavathi)

N180509 (P. Naga lakshmi)

N180257 (Sk. Hussain)

N180086(M. Thrisali)

Under the Esteem Guidance of

Dr. Sadu Chiranjeevi

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGETECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RGUKT-Nuzvid, Eluru Dist – 521202

DECLARATION

We certify that

a. The work contained in this report is original and has been done by us under the guidance of my supervisor.

b. The work has not been submitted to any other Institute for any degree or diploma.

c. We have followed the guidelines provided by the Institute in preparing the report.

d. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever we have used materials (data, theoretical analysis, figures, and text) from other sources, we have given due credit to them by citing them in the text of the report and giving their details in the references. Further, we have taken permission from the copyright owners of the sources, whenever necessary.

N180114 (K. Sindhuja)

N180117(A. Pushpavathi)

N180509 (P. Naga lakshmi)

N180257 (Sk. Hussain)

N180086(M. Thrisali)

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

RGUKT-Nuzvid, Eluru Dist – 521202

CERTIFICATE

This is to certify that the Dissertation report entitled, **"Image Generation using Generative Adversarial Networks (GANs)"**submitted by **K.Sindhuja, A.Pushpavathi, P.Naga lakshmi, Sk.Hussain, M.Thrisali** to Rajiv Gandhi university of Knowledge Technologies, Nuzvid, India, is a record of bonafide Project work carried out by us under my/our supervision and guidance and is worthy of consideration for the fulfillment of mini-project of Bachelor of Technology in computer Science and Engineering of the Institute.

Dr. Sadu Chiranjeevi	**Examiner**
Project Supervisor Examiner	Project Examiner
Faculty Dept. of CSE	Faculty Dept. of CSE
RGUKT Nuzvid	**RGUKT Nuzvid**

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide **Dr. Sadu Chiranjeevi** for his exemplary guidance, monitoring and constant encouragement to us throughout this semister. We shall always cherish the time spent with him during the course of this work due to the invaluable knowledge gained in the field of Machine Learning.

We are extremely grateful for the confidence bestowed in us and entrusting our project entitled **"Image Generation using Generative Adversarial Networks".**

We express gratitude to our HOD sir, Dept. of CSE and other faculty members and our beloved seniors for being source of inspiration and constant encouragement which helped us in completing the project successfully.

N180114(K. Sindhuja)
N180117(A. Pushpavathi)
N180509 (P. Naga lakshmi)
N180257 (Sk. Hussain)
N180086(M. Thrisali)

CONTENTS

ABSTRACT

Generative Adversarial Networks are a kind of artificial intelligence algorithm designed to solve the generative modelling problem [1]. The goal of a generative model is to study a collection of training examples and learn the probability distribution that generated them. Generative Adversarial Networks (GANs) are then able to generate more examples from the estimated probability distribution. Generative models based on deep learning are common, but GANs are among the most successful generative models (especially in terms of their ability to generate realistic high-resolution images). GANs have been successfully applied to a wide variety of tasks (mostly in research settings) and synthetic medical image generation [4], but continue to present unique challenges and research opportunities because they are based on game theory while most other approaches to generative modeling are based on optimization.

Our project is to study a collection of training examples, then learn to generate more examples that come from the same probability distribution. GANs learn to do this without using an explicit representation of the density function. One advantage of the GAN framework is that it may be applied to models for which the density function is computationally intractable. The samples shown here are all samples from the CelebA dataset [6], 8 including the ones labeled "model samples." We use actual CelebA data to illustrate the goal that a hypothetical perfect model would attain.

INTRODUCTION

Most current approaches to developing artificial intelligence are based primarily on machine learning. The most widely used and successful form of machine learning to date is supervised learning. Supervised learning algorithms are given a dataset of pairs of example inputs and example outputs. They learn to associate each input with each output and thus learning a mapping from input to output examples. The input examples are typically complicated data objects like images [**2**], natural language sentences, or audio wave forms, while the output examples are often relatively simple.

Supervised learning is often able to achieve greater than human accuracy after the training process is complete, and thus has been integrated into many products and services. Unfortunately, the learning process itself still falls far short of human abilities. Supervised learning by definition relies on a human supervisor to provide an output example for each input example. Worse, existing approaches to supervised learning often require millions of training examples to exceed human performance, when a human might be able to learn to perform the task acceptably from a very small number of examples.

In order to reduce both the amount of human supervision required for learning and the number of examples required for learning, many researchers today study unsupervised learning, often using generative models. In this overview paper, we describe one particular approach to unsupervised learning via generative modeling called generative adversarial networks [2]. We briefly review applications of GANs and identify core research problems related to convergence in games necessary to make GANs a reliable technology.

BACKGROUND AND RELATED WORKS

GANs are a new framework for generative models that were proposed in 2014.GANs consist of two neural networks: a generator and a discriminator. The generator tries to produc e realistic samples that can fool the discriminator, while the discriminator attempts to distingu ish real samples from generated ones. GANs have shown promising results encryption schem es that enable data owners to share their encrypted data with others in a secure manner. Howe ver, the challenge remains in how to generate realistic and diverse synthetic data for applicati ons such as data augmentation and privacy protection. Generative Adversarial Networks (GA Ns) offer a promising solution to this challenge.

One of the potential applications of GANs in data management is data augmentation. With the ability to generate diverse and realistic synthetic data, GANs can be used to augment existing datasets, enabling more robust and accurate machine learning models. In addition, GANs can also be used for privacy protection, by generating synthetic data that preserve the statistical properties of the original data while masking sensitive information.

However, GANs also face challenges in data management applications. For example, generating diverse and realistic synthetic data requires large amounts of training data, which may not always be available. In addition, GANs may suffer from mode collapse, where the generator produces a limited set of outputs that do not cover the full range of possible samples.

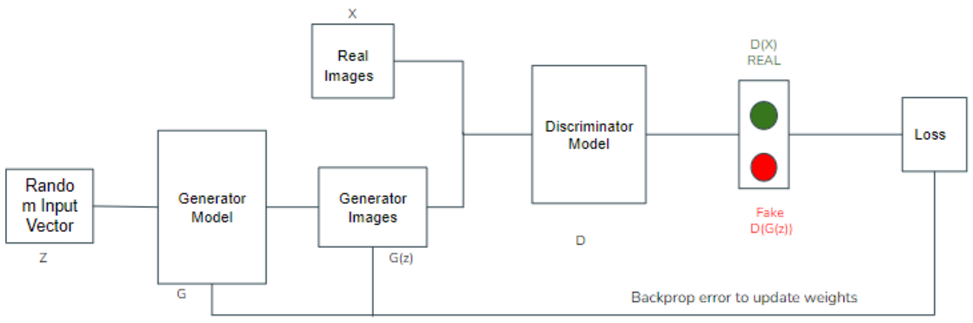
Despite these challenges, GANs have great potential in data management applications in the cloud computing era. As more and more data are uploaded and stored in public clouds, GANs can offer a powerful tool for data augmentation and privacy protection, enabling more secure and efficient data sharing in the cloud.

METHODOLOGIES

The goal of supervised learning is relatively straight forward to specify, and all supervised learning algorithms have essentially the same goal: learn to accurately associate new input examples with the correct outputs. Unsupervised learning is a less clearly defined branch of machine learning, with many different unsupervised learning algorithms pursuing many different goals.

Generative Adversarial Networks (GANS):

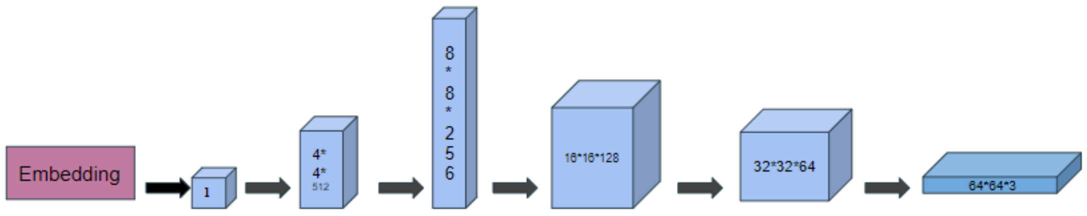
The generative adversarial networks (GANs), introduced by Ian J. Goodfellow and collaborators in 2014 [3], is one of the recently developed approaches to ‘generative model’ using a flexible unsupervised deep learning architecture [2]. Generative model is an important task carried out using unsupervised learning to automatically generate and learn the patterns in input data so that the model can be utilized for generating new examples (output) that could have been drawn using the original dataset. The ability of GAN to generate new content makes it more popular and useful in real-life image generation.



Generator:

Generative Adversarial Networks (GANs), the generator refers to a neural network component of the model. The generator network takes random noise as input and generates synthetic data samples that resemble the training data it was trained on. The goal of the generator is to produce samples that are indistinguishable from real data.

The generator takes a low-dimensional noise vector as input and gradually increases the spatial dimensions and decreases the number of channels to generate realistic images. It starts with a low-resolution representation and progressively up samples the features until it reaches the desired output image dimensions.



Discriminator:

The discriminator is trained to classify input samples as either real or generated. It takes input data, which can be real or generated samples, and produces an output that represents the probability of the input being real. The output is typically a single scalar value, with values closer to 1 indicating high confidence that the input is real, and values closer to 0 indicating high confidence that the input is generated.

The discriminator network progressively reduces the spatial dimensions and increases the number of channels, the generator network performs the opposite operations. As the discriminator receives input images, it aims to distinguish between real and generated/fake images by learning to recognize patterns and features indicative of real data. By gradually reducing the spatial dimensions and increasing the number of channels, the discriminator is able to effectively analyze the input at different levels of detail.

Calculations:

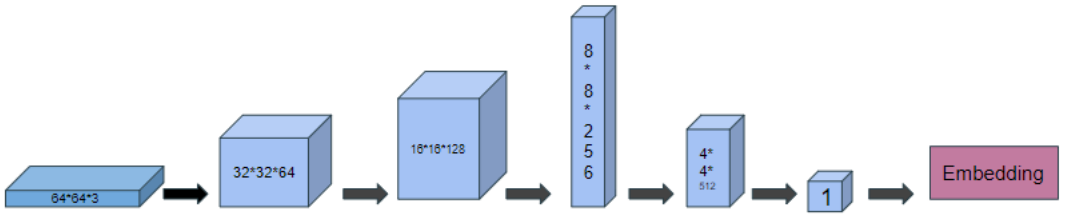
output_size = [(input size - kernel size + 2 * padding) / stride] + 1

Where:

- input size: The input spatial dimension (width or height) of the feature map.
- kernel size: The size of the convolutional kernel (filter).
- padding: The amount of zero-padding applied to the input feature map.
- stride: The stride or step size used for the convolution operation.

- Initial Input:
  - Input Dimensions: 64x64 pixels
  - Number of Channels: 3
- After the first convolutional layer:
  - Output Dimensions: (64 - 4 + 2*1) / 2 + 1 = 32
  - Number of Channels: 64
- After the second convolutional layer:
  - Output Dimensions: (32 - 4 + 2*1) / 2 + 1 = 16
  - Number of Channels: 128

- After the third convolutional layer:
    - Output Dimensions: (16 - 4 + 2*1) / 2 + 1 = 8
    - Number of Channels: 256
- After the fourth convolutional layer:
    - Output Dimensions: (8 - 4 + 2*1) / 2 + 1 = 4
    - Number of Channels: 512
- After the final convolutional layer:
    - Output Dimensions: (4 - 4 + 2*0) / 1 + 1 = 1
    - Number of Channels: 1

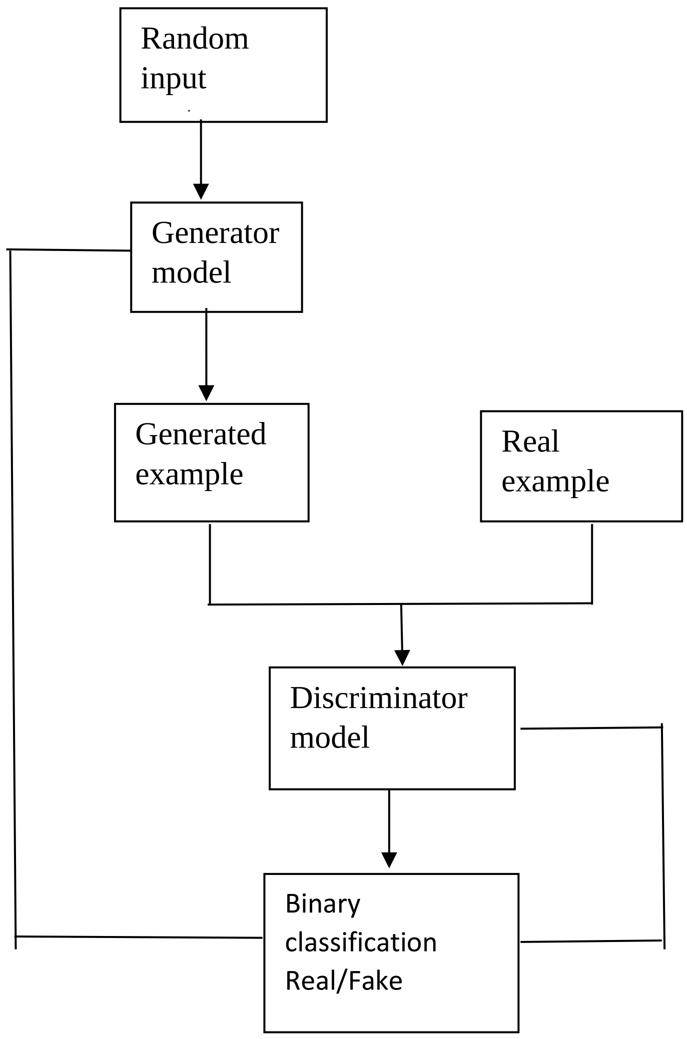


The purpose of having 1 channel in the final layer of the discriminator is to produce a single scalar value or score for each input image. This score represents the probability or confidence that the input image is real or fake. The discriminator uses this output to distinguish between real and generated/fake images.

Working of gan:

Training GANs involves training both a generator network and a discriminator network. The process involves both real data drawn from a dataset and fake data created continuously by the generator throughout the training process. The discriminator is trained much like any other classifier defined by a deep neural network [3]. As shown on the left, the discriminator is shown data from the training set. In this case, the discriminator is trained to assign data to the “real” class. As shown on the right, the training process also involves fake data. The fake data is constructed by first sampling a random vector z from a prior distribution over latent variables of the model. The generator is then used to produce a sample x = G(z). The function G is simply a function represented by a neural network that transforms the random, unstructured z vector into structured data, intended to be statistically indistinguishable from the training data. The discriminator then classifies this fake data. The discriminator is trained to assign this data to the “fake” class. The backpropagation algorithm makes it possible to use the derivatives of the discriminator’s output with respect to the discriminator’s input to train the generator. The generator is trained to fool the discriminator, in other words, to make the discriminator assign its input to the “real” class. The training process for the discriminator is thus much the same as for any other binary classifier with the exception that the data for the “fake” class comes from a distribution that changes constantly as the generator learns rather than from a fixed distribution. The learning process for the generator is somewhat unique, because it is not given specific targets for its output, but rather simply given a reward for producing outputs that fool its opponent.

Flow chart:



Hyperparameters:

Hyperparameters in Deep Convolutional Generative Adversarial Networks (DCGANs) are the adjustable settings that govern the behaviour and performance of the model during training.

- Learning Rate (lr)=0.00025
- Batch_Size=128
- Number of Epochs=60

- Latent Space Dimension (latent_size) =128
- Activation Functions (LeakyReLu. ntAdam optimizer)

Loss function:

The Generator Model ***G*** takes a random input vector ***z*** as an input and generates the images ***G(z).*** These generated images along with the real images ***x*** from training data is then fed to the Discriminator Model ***D***. The Discriminator Model then classifies the images as real or fake. Then, we have to measure the loss and this loss has to be back propagated to update the weights of the Generator and the Discriminator[2].

When we are training the Discriminator, we have to freeze the Generator and back propagate errors to only update the Discriminator.When we are training the Generator, we have to freeze the Discriminator and back propagate errors to only update the Generator.Thus, the Generator Model and the Discriminator Model getting better and better at each epoch.

Let us introduce some notations to understand the loss function of the GANs.

G	Generator model
D	Discriminator model
z	Random noise (Fixed size input vector)
x	Real image
G(z)	Image generated by generator (Fake image)
P_{data}	Probability distribution of real images
$P_z(z)$	Probability distribution of fake images
D(G(z))	Discriminator's output when the generated image is an input
D(x)	Discriminator's output when the real image is an input

$$\boxed{\textbf{Min max V (D, G) = min max } (E_{x\sim p_{data(x)}}[\log D(x)] + E_{z\sim p_{z(z)}}[\log(1-D(G(z)))])}$$

The Generator wants to minimize the *V (D, G)* whereas the Discriminator wants to maximize the *V (D, G)*. Let us understand both terms:

1. $E_{x\sim pdata(x)}$ [log D(x)]: Average log probability of D when real image is input.

2. $E_{z\sim pz(z)}$ [log (1 – D(G(z)))]: Average log probability of D when the generated image is input.

Discriminator loss:

While training discriminator the label of data coming from $P_{data}(x)$ is y=1 (real data) and y^=D(x) substituting this in above loss function we get,

$$L(D(x),1) = \log(D(x)) \text{ -----------> } (1)$$

And for data coming from generator, the label is y=0(fake data) and y^=D(g(z)). So, in this case,

$$L(D(G(z)),0) = \log(1-D(G(z))) \text{ ---------> } (2)$$

Now, the objective of the discriminator is to correctly classify the fake and real dataset. For this, equations (1) and (2) should be maximized and final loss function for the discriminator can be given as,

$$L^{(D)}=\max[\log(D(x)+\log(1-D(G(z)))] \text{ ------------> } (3)$$

Generator loss:

Here, the generator is competing against discriminator. So, it will try to minimize the equation (3) and loss function is given as,

$$L^{(G)} = \min[\log(D(x)) + \log(1-D(G(z)))] \text{ ---------> } (4)$$

Combined loss function:

We can combine equations (3) and (4) and write as,

$$L=\min \max [\log(D(x)) + \log(1-D(G(z)))] \text{ ----------> } (5)$$

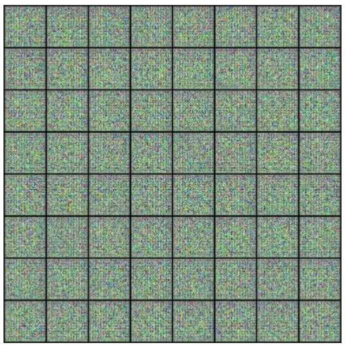
Remember that the above loss function is valid only for a single data point, to consider entire dataset we need to take the expectation of the above equation as

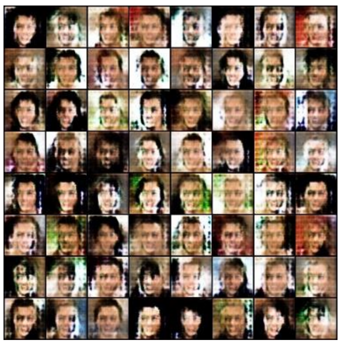
$$\text{Min max } V (D, G) = \min \max (E_{x\sim p_{data(x)}}[\log D(x)] + E_{z\sim p_{z\,(z)}}[\log(1-D(G(z)))]) \text{ -------}(6)$$

ANALYSIS

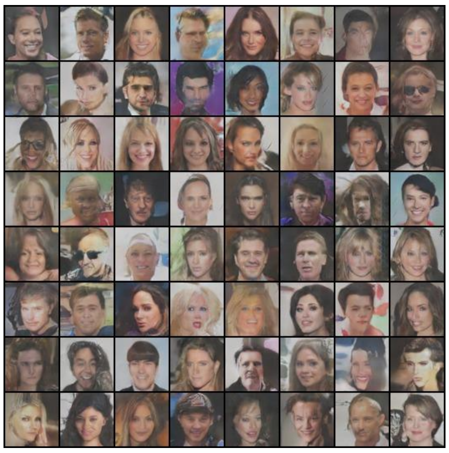
- At the beginning of training (0 epoch), the generator's performance is usually poor, and the generated samples are often far from being realistic. Consequently, the generator loss is initially high.
- As training progresses, the generator learns to generate more plausible samples that resemble the real data. The generator loss tends to decrease over time as the generator becomes more skilled at fooling the discriminator. By epoch 60, the generator loss should ideally be significantly lower than at the start of training, indicating improved performance.
- On the other hand, the discriminator loss function measures how well the discriminator distinguishes between real and generated samples. At 0 epoch, the discriminator is untrained and randomly predicts the labels for both real and generated samples. As a result, the discriminator loss is relatively high at the beginning.
- During training, the discriminator learns to better differentiate between real and generated samples, aiming to minimize its loss. As the generator improves, the discriminator faces a more challenging task, which can lead to increased discriminator loss. However, the discriminator should still be able to adapt and improve its discrimination abilities

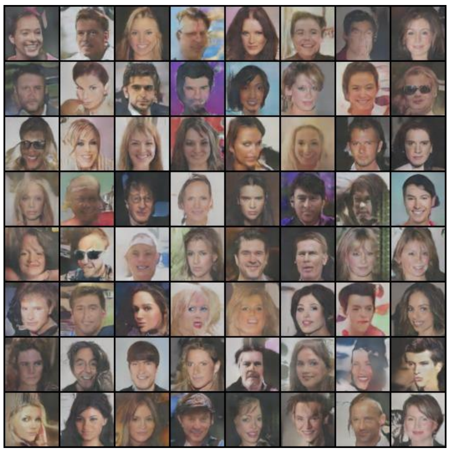
RESULT


Epoch-1

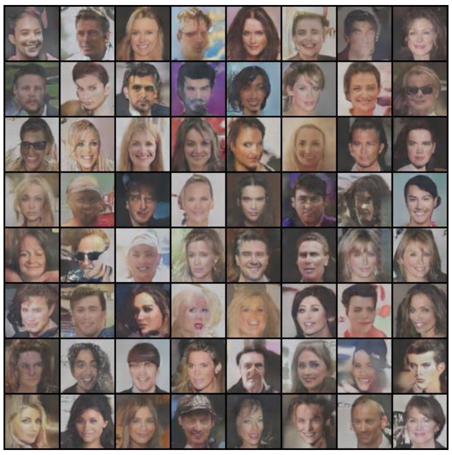

Epoch-2

Epoch-3


Epoch-57


Epoch-58

Epoch-59


Epoch-60

S_No	**Epoch_No**	**Loss_g**	**Loss_d**	**Real_score**	**Discriminator_score** **D_S=(r_s+f_s)/2**	**Fake_score** **(or)** **Generator_score**
1	1	4.3471	0.7288	0.8157	0.5831	0.3521
2	2	1.4118	0.8048	0.5332	0.28225	0.0313
3	3	0.7902	2.7108	0.1223	0.0648	0.0073
4	4	4.0678	0.2505	0.8110	0.4171	0.0232
5	5	3.0842	0.5418	0.8750	0.57865	0.2823
6	55	5.2828	0.1126	0.9645	0.51185	0.0592
7	56	3.4084	0.6058	0.9205	0.61395	0.3074
8	57	5.8029	0.2901	0.9789	0.5795	0.1801
9	58	4.9250	0.1055	0.9359	0.4821	0.0283
10	59	5.9607	0.1349	0.9953	0.5508	0.1063
11	60	3.6739	0.3585	0.9319	0.567	0.2021

CONCLUSION

In conclusion, this project aimed to develop and train a GAN model to generate high-quality images that are visually appealing and indistinguishable from real images. The project focused on the image generation aspect of GANs and involved tasks such as collecting and preprocessing image data, developing and training the GAN model, testing and fine-tuning the model, and integrating it into a user-friendly interface for image generation. This also provided the ability to customize the GAN model for specific image generation tasks and evaluated its performance based on image quality, training time, and computational resources used. The results of the project showed that a GAN model can be trained to generate high-quality images with a user-friendly interface that allows customization of various parameters. This system can have various applications in fields such as gaming, art, and e-commerce. This project has laid the foundation for further research and development in this area, with the ultimate goal of creating highly realistic and customizable images for a variety of applications.

REFERENCES

1. Ian Goodfellow et al. in their 2014 paper, https://arxiv.org/abs/1406.2661

2. Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, Université de Montréal Generative adversarial networks (acm.org)

3.
   Mayank Vadsola https://towardsdatascience.com/the-math-behind-gans-generative-adversarial-networks-3828f3469d9c

4. Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio

   https://dl.acm.org/doi/pdf/10.1145/3422622

5. zohaib ahma, Chughtai chughtia, Rizwan Malik, Sidra chugtai

   http://ijcis.com/index.php/IJCIS/article/view/31

6. CelebA dataset of size 100k from Kaggle website
   https://www.kaggle.com/datasets/jessicali9530/celeba-dataset