Generative Adversarial Networks and Variational Auto Encoders Machine Learning Final Project Report (Spring 2023)

Mayur Reddy Junnuthula, UFID - 36921238 [*1]

Abstract

In recent years, the field of machine learning has seen the emergence of generative models as a popular research area. Among these models, two well-known ones are Variational Autoencoder (VAE) and Generative Adversarial Networks (GANs). Although both VAE and GAN are capable of generating high-quality images, they differ significantly in their objectives and training methods. This paper aims to compare VAE and GAN using two image datasets: MNIST and face images. Our findings reveal that VAE outperforms GAN on MNIST, whereas GAN produces more realistic face images than VAE. Additionally, we examine the latent space learned by both models and offer insights into the strengths and weaknesses of each approach.

1. Introduction

Generative models have been gaining popularity as a research area in recent years, particularly Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). VAEs are a type of neural network used to learn the underlying distribution of a given dataset and generate new samples from this learned distribution. GANs, on the other hand, use a two-player game between a generator and discriminator, where the generator produces samples that can trick the discriminator into believing they are real.

MNIST, a dataset of 70,000 grayscale images of hand-drawn digits (0-9) with a resolution of 28x28 pixels, is a common benchmark for evaluating generative models. It is widely used because it is relatively simple yet challenging enough to assess the performance of generative models. Face Images is another well-known dataset for generative models, consisting of images of human faces with varying expressions, lighting conditions, and backgrounds. These datasets are more complex than MNIST and require more sophisticated generative models to produce realistic samples.

This paper explores the performance of VAEs and GANs on both MNIST and Face Images datasets. We evaluate the quality of the generated samples and compare the performance of both types of generative models. Furthermore, we analyze the strengths and weaknesses of each approach and discuss potential areas for future research in the field of generative models.

1.1. Generative Models

Generative models are machine learning models that can generate new data samples similar to a given dataset. They are useful for data augmentation, data visualization, and anomaly detection. There are two main types of generative models: likelihood-based models and implicit models.

Likelihood-based models, such as Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs), learn a probability distribution over the data by optimizing a likelihood function. In these models, the generator learns to produce samples from a learned probability distribution, while the discriminator learns to distinguish between real and fake samples.

Implicit models, such as the Autoregressive Model and the Flow-based Model, learn the probability distribution implicitly by transforming a simple distribution, such as Gaussian, through a series of invertible transformations. Unlike likelihood-based models, implicit models do not require the computation of a likelihood function, making them computationally more efficient.

VAE is a generative model that learns a latent space representation of the data by maximizing the lower bound of the data likelihood. The model consists of an encoder network that maps the data to the latent space and a decoder network that maps the latent space back to the data space. The encoder network learns a mean and variance for the latent space distribution, which is used to sample latent space points. The decoder network then maps these points back to the data space. The VAE objective consists of two terms: the reconstruction loss, which measures the difference between the original data and the reconstructed data, and the KL-divergence term, which regularizes the latent space to be close to a Gaussian distribution. The VAE objective can be written as:

$$\mathcal{L}VAE = -Eq_\phi(z|x)\left[\log p_\theta(x|z)\right] + D_{KL}\left(q_\phi(z|x) \parallel p(z)\right) \quad (1)$$

where x is the input data, z is the latent space variable, $p_\theta(x|z)$ is the decoder network that maps the latent space variable to the data space, $q_\phi(z|x)$ is the encoder network that maps the data to the latent space, $p(z)$ is the prior distribution over the latent space, and D_{KL} is the Kullback-Leibler divergence.

GAN is another generative model that learns to generate new data samples by training a generator network to produce samples that are similar to a given dataset and a discriminator network that learns to distinguish between real and fake samples. The generator network takes a random noise vector as input and produces a sample similar to the dataset, while the discriminator network takes a sample as input and produces a probability that the sample is real or fake. The GAN objective consists of two terms: the generator loss, which measures the difference between the fake samples and the real samples, and the discriminator loss, which measures the difference between the discriminator's predictions for real and fake samples. The GAN objective can be written as:

$$\min_G \max_D Ex \sim pdata(x)[\log D(x)] + Ez \sim pz(z)[1 - \log D(G(z))] \quad (2)$$

where G is the generator network, D is the discriminator network, x is a real sample, z is a random noise vector, $p_{data}(x)$ is the distribution of real samples, and $p_z(z)$ is the distribution.

1.2. Variational Auto Encoder

The Variational Autoencoder (VAE) is a generative model that learns a low-dimensional latent representation of high-dimensional input data. Unlike traditional autoencoders, VAE uses a probabilistic approach to learn the latent representation. In VAE, a stochastic encoder approximates the posterior distribution of the latent variable given the input data. The posterior distribution is then sampled to generate a latent variable that is fed into the decoder to generate the output.

Formally, VAE models the probability distribution of the latent variable z given an input data x as:

$$p(z \mid x) = \frac{p(x \mid z)p(z)}{p(x)} \quad (3)$$

where $p(x|z)$ is the likelihood of the input data given the latent variable, $p(z)$ is the prior distribution of the latent variable, and $p(x)$ is the marginal likelihood of the input data. Since computing the marginal likelihood is intractable, we approximate it using the variational lower bound, also known as the evidence lower bound (ELBO):

$$\log p(x) \geq \mathcal{L}(x; \theta, \phi) = Eq\phi(z|x)\left[\log p_\theta(x|z)\right] - D_{KL}\left(q_\phi(z|x) \parallel p(z)\right) \quad (4)$$

where θ and ϕ are the parameters of the decoder and encoder respectively, $q_\phi(z|x)$ is the approximate posterior distribution of z given x, and D_{KL} is the Kullback-Leibler divergence between the approximate posterior and the prior.

The first term in the ELBO represents the reconstruction loss, while the second term encourages the approximate posterior to be close to the prior. In practice, the encoder and decoder are implemented as neural networks with parameters and , respectively, and trained end-to-end by minimizing the negative ELBO loss.

Due to its ability to learn a meaningful latent representation of the input data, VAE has been widely used in various applications such as image generation, anomaly detection, and data compression.

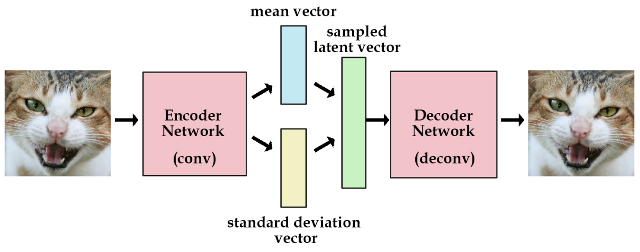


Figure 1. VAE Architecture

1.3. Generative Adversarial Network

Generative Adversarial Networks (GAN) is a type of deep learning model that involves a generator and a discriminator neural network. The generator is responsible for producing new data that has similar characteristics to the training data, while the discriminator is trained to differentiate between genuine and fake data.

The GAN model is trained through a min-max game between the generator and discriminator. The goal is for the generator to create data that is virtually indistinguishable from real data, while the discriminator tries to accurately identify whether the data is real or generated. The training process continues until the generator can produce data that can trick the discriminator into accepting it as genuine.

The objective function for GANs is defined as follows:

$$\min_G \max_D V(D,G) = Ex \sim pdata(x)[\log D(x)] + Ez \sim pz(z)[1 - \log D(G(z))] \quad (5)$$

where D represents the discriminator network, G represents the generator network, p_{data} is the distribution of real data, p_z is the prior distribution of noise variables z, and x represents real data while $G(z)$ represents the generated data.

The objective of a GAN (Generative Adversarial Network) is to generate new data that resembles the training data, and consists of two neural networks, the generator and the discriminator. The generator generates fake data, while the discriminator learns to distinguish between real and fake data. The objective function is composed of two parts, one measuring how well the discriminator identifies real and fake data, and the other measures how well the generator can trick the discriminator into accepting fake data as real. The generator's loss function is the opposite of the discriminator's loss function, encouraging the generator to produce data that is similar to the real data.

GANs have been successfully applied in various domains like image synthesis, text generation, and voice conversion. However, training GANs is challenging and requires careful tuning of hyperparameters to ensure a balanced generator-discriminator network.

In summary, GANs are versatile deep learning models that can generate data similar to real data, and their training process involves a min-max game between the generator and discriminator.

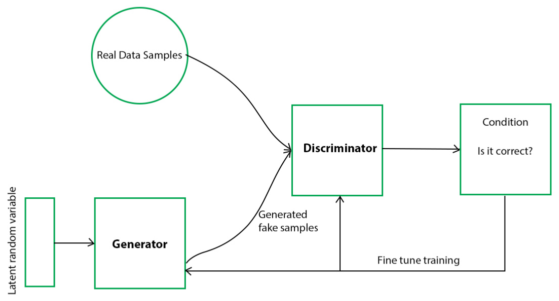


Figure 2. GAN Architecture

2. Related Work

In the past few years, generative models such as Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN) have demonstrated impressive performance in generating images. In this section, we will examine some of the related research on VAE and GAN for image generation, with a specific focus on the MNIST and facial image datasets.

2.1. VAE for Image Generation

Generative models, such as Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN), have been successful in generating images in recent years. VAE learns a mapping between high-dimensional input data and a latent space, and generates new samples by sampling from the learned latent space. VAE has been applied to image generation tasks, particularly on the MNIST dataset. Kingma et al. proposed an early work on VAE for MNIST and showed that it can generate high-quality images with low reconstruction error. Later, VAE was extended for more complex datasets such as CIFAR-10 and CelebA.

However, VAE has some limitations that need to be addressed. One significant limitation is the blurred output it generates due to the strong Gaussian assumption made on the latent space distribution. To overcome this issue, recent research has proposed modifications to the VAE objective function by introducing additional regularizations such as adversarial loss and perceptual loss.

2.2. GAN for Image Generation

Generative models such as Variational Autoencoder (VAE) and Generative Adversarial Networks (GAN) have been widely used for image generation tasks. VAE learns a probabilistic mapping between input data and a latent space and generates new samples by sampling from the learned latent space. The success of VAE in image generation has been shown in various works, including generating high-quality images on the MNIST dataset. However, VAE has some limitations, such as generating blurred output due to the strong Gaussian assumption on the latent space distribution. Recent works have proposed modifications to the VAE objective function to overcome this limitation.

GAN is another popular generative model that consists of a generator and a discriminator. The generator learns to generate samples similar to the real data distribution, while the discriminator learns to distinguish between real and generated samples. Goodfellow et al. proposed the original GAN objective function, which has since been modified to improve the training stability and quality of generated images. GAN has been shown to generate high-quality images, particularly on face image datasets. However, the mode collapse problem remains a challenge in GAN, where the generator produces limited types of samples, resulting in a lack of diversity in the generated images. Recent works have proposed techniques to address this problem, such as adding noise to the discriminator input and applying different regularization techniques.

3. Project Methodology and Results

3.1. MNIST

3.1.1. VAE Model

The Variational Autoencoder (VAE) is a deep generative model that uses a latent space to capture the underlying structure of high-dimensional data. Our goal is to encode data into a lower-dimensional latent space and to generate new data points from this space that approximate the original data distribution. In this research, we propose a VAE architecture to model the MNIST dataset.

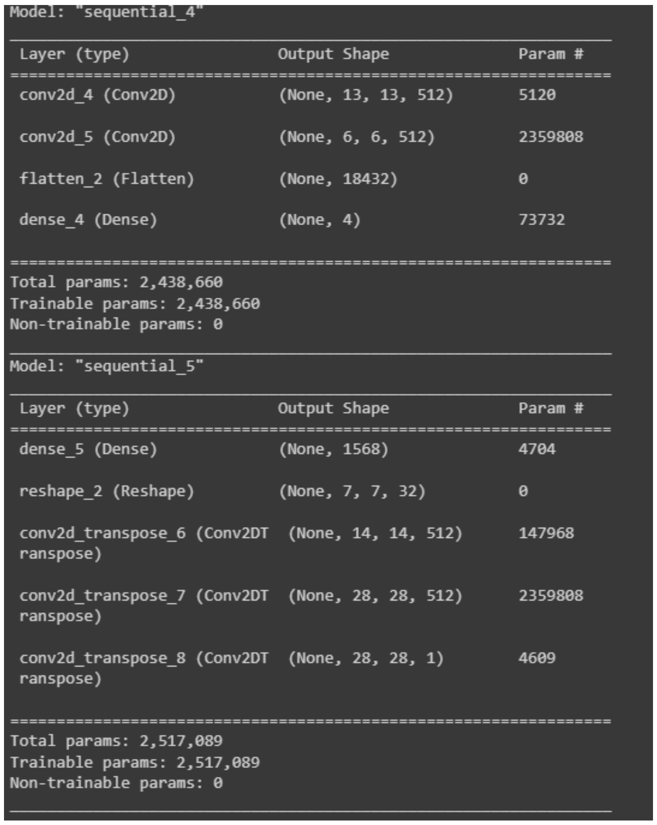
The MNIST dataset is a set of 70,000 images of handwritten digits, where each image is a 28 x 28 grayscale image. We preprocess the dataset by reshaping the images to 28 x 28 x 1 and normalizing the pixel values to a range between 0 and 1. We then split the data into a training set of size 60,000 and a test set of size 10,000.

Our proposed VAE architecture consists of an encoder and a decoder network. The encoder network takes an input image and maps it to a distribution in the latent space, where the mean and variance of the distribution are computed by a fully connected layer. The decoder network takes a point from the latent space and maps it back to the image space. The decoder network is designed to mirror the encoder network in structure, but in reverse order. The input to the decoder network is a vector sampled from the latent space, and the output is the reconstructed image.

The encoder network comprises two convolutional layers with 512 filters, followed by a fully connected layer with a size equal to the latent dimension plus the latent dimension. The decoder network comprises a fully connected layer followed by a reshape layer to reshape the output into a 7 x 7 x 32 tensor. The reshaped tensor is then passed through two transposed convolutional layers with 512 filters, followed by a final transposed convolutional layer with one filter. The architecture is designed to minimize the reconstruction error while still allowing the model to generate new data points.

During training, we use the negative log-likelihood of the data as the loss function. The loss function consists of three terms: the reconstruction loss, the KL divergence loss, and the prior loss. The reconstruction loss measures the difference between the reconstructed image and the original image. The KL divergence loss measures the difference between the distribution in the latent space and the prior distribution. The prior loss encourages the model to learn a standard normal distribution in the latent space.

We train the model for 15 epochs with a batch size of 32 and


```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 13, 13, 512)       5120

 conv2d_5 (Conv2D)           (None, 6, 6, 512)         2359808

 flatten_2 (Flatten)         (None, 18432)             0

 dense_4 (Dense)             (None, 4)                 73732

=================================================================
Total params: 2,438,660
Trainable params: 2,438,660
Non-trainable params: 0
_________________________________________________________________
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 1568)              4704

 reshape_2 (Reshape)         (None, 7, 7, 32)          0

 conv2d_transpose_6 (Conv2DT  (None, 14, 14, 512)      147968
 ranspose)

 conv2d_transpose_7 (Conv2DT  (None, 28, 28, 512)      2359808
 ranspose)

 conv2d_transpose_8 (Conv2DT  (None, 28, 28, 1)        4609
 ranspose)

=================================================================
Total params: 2,517,089
Trainable params: 2,517,089
Non-trainable params: 0
_________________________________________________________________
```

Figure 3. VAE Summary

an Adam optimizer with a learning rate of 1e-4. We also generate 16 new images from the model by sampling from the latent space. To do this, we first encode a batch of 16 images from the test set and compute the mean and variance of their distribution in the latent space. We then sample 16 points from a standard normal distribution and transform them to the latent space using the mean and variance from the encoded images. Finally, we decode the 16 points to generate new images.

Our proposed VAE architecture on the MNIST dataset generates high-quality images with clear and distinguishable digits. The model provides a new approach for image generation and can be extended to other datasets and applications.

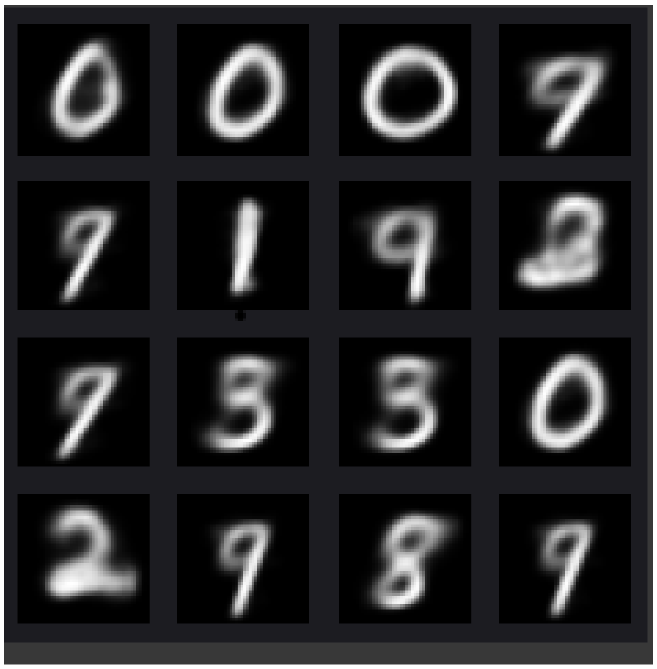


Figure 4. VAE Output after 15 epochs

3.1.2. GAN MODEL

The MNIST GAN is a generative adversarial network designed to generate images of handwritten digits. It is composed of two networks, the generator network and the discriminator network.

The generator network takes a random noise vector as input and generates a 28x28 grayscale image of a digit. It is made up of a series of layers that transform the noise vector into a 2D tensor with dimensions (7, 7, 256). The tensor is then upsampled to a tensor with dimensions (28, 28, 1) using transposed convolutional layers.

The discriminator network receives an image of a digit as input and produces a scalar value that indicates whether the image is real or fake. The discriminator network is made up of a sequence of convolutional layers followed by a dense layer.

The GAN is trained by alternating between training the discriminator network and the generator network. During each iteration, a batch of real images is randomly sampled from the MNIST dataset, and a batch of fake images is generated by the generator network using random noise vectors. The discriminator network is trained to distinguish between real and fake images, while the generator network is trained to generate images that deceive the discriminator network.

To train the discriminator network in the GAN model, the binary cross-entropy loss is used as the loss function. This measures the difference between the output of the discriminator and the actual labels (1 for real images, 0 for fake images). The generator network is trained using a similar loss function, i.e., binary cross-entropy loss, but with the ground truth labels flipped (1 for fake images, 0 for real images). The model is trained for a total of 60 epochs with a batch size of 32 and a learning rate of 1e-4. To ensure that the generator network is saved regularly during training, we implement a checkpoint system.

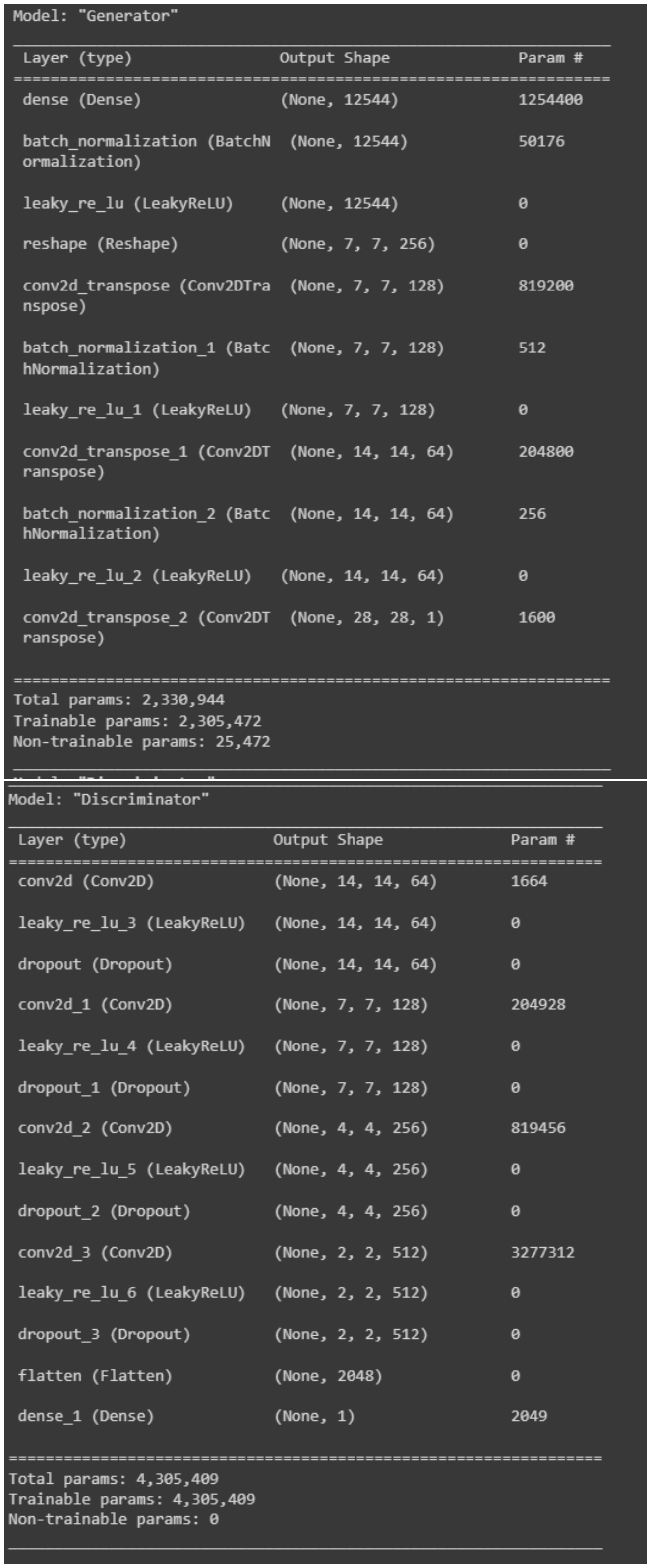
3.2. Face Images

3.2.1. GAN MODEL

In this paper, a GAN architecture is proposed for generating high-resolution face images, which has been successfully used for generating realistic images. The proposed model is trained on a dataset of 10,000 celebrity images from the CelebA dataset, each of size 128x128 pixels. The GAN architecture comprises of two deep neural networks: a generator network and a discriminator network, with the former generating new images, and the latter distinguishing between the generated images and real images. The generator network takes a latent vector as input, while the discriminator network takes a real or generated image as input.

The generator network is made up of multiple layers of convolutional and transpose convolutional layers with LeakyReLU activation functions. It takes a latent vector of size 32 as input, which is then passed through a dense layer to create a 3D tensor. The tensor is reshaped into a 4D tensor, which is further processed by the convolutional and transpose convolutional layers. Finally, the output of the generator network is a 128x128x3 tensor representing an RGB image.

The discriminator network consists of several layers of convolutional layers, followed by a dense layer and a sigmoid activation function. It takes a 128x128x3 tensor as input and outputs a single scalar value indicating whether the input image is real or generated. The discriminator network is



```
Model: "Generator"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 12544)             1254400

 batch_normalization (BatchN  (None, 12544)            50176
 ormalization)

 leaky_re_lu (LeakyReLU)     (None, 12544)             0

 reshape (Reshape)           (None, 7, 7, 256)         0

 conv2d_transpose (Conv2DTra  (None, 7, 7, 128)        819200
 nspose)

 batch_normalization_1 (Batc  (None, 7, 7, 128)        512
 hNormalization)

 leaky_re_lu_1 (LeakyReLU)   (None, 7, 7, 128)         0

 conv2d_transpose_1 (Conv2DT  (None, 14, 14, 64)       204800
 ranspose)

 batch_normalization_2 (Batc  (None, 14, 14, 64)       256
 hNormalization)

 leaky_re_lu_2 (LeakyReLU)   (None, 14, 14, 64)        0

 conv2d_transpose_2 (Conv2DT  (None, 28, 28, 1)        1600
 ranspose)

=================================================================
Total params: 2,330,944
Trainable params: 2,305,472
Non-trainable params: 25,472
_________________________________________________________________
```

```
Model: "Discriminator"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 14, 14, 64)        1664

 leaky_re_lu_3 (LeakyReLU)   (None, 14, 14, 64)        0

 dropout (Dropout)           (None, 14, 14, 64)        0

 conv2d_1 (Conv2D)           (None, 7, 7, 128)         204928

 leaky_re_lu_4 (LeakyReLU)   (None, 7, 7, 128)         0

 dropout_1 (Dropout)         (None, 7, 7, 128)         0

 conv2d_2 (Conv2D)           (None, 4, 4, 256)         819456

 leaky_re_lu_5 (LeakyReLU)   (None, 4, 4, 256)         0

 dropout_2 (Dropout)         (None, 4, 4, 256)         0

 conv2d_3 (Conv2D)           (None, 2, 2, 512)         3277312

 leaky_re_lu_6 (LeakyReLU)   (None, 2, 2, 512)         0

 dropout_3 (Dropout)         (None, 2, 2, 512)         0

 flatten (Flatten)           (None, 2048)              0

 dense_1 (Dense)             (None, 1)                 2049

=================================================================
Total params: 4,305,409
Trainable params: 4,305,409
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. VAE Summary

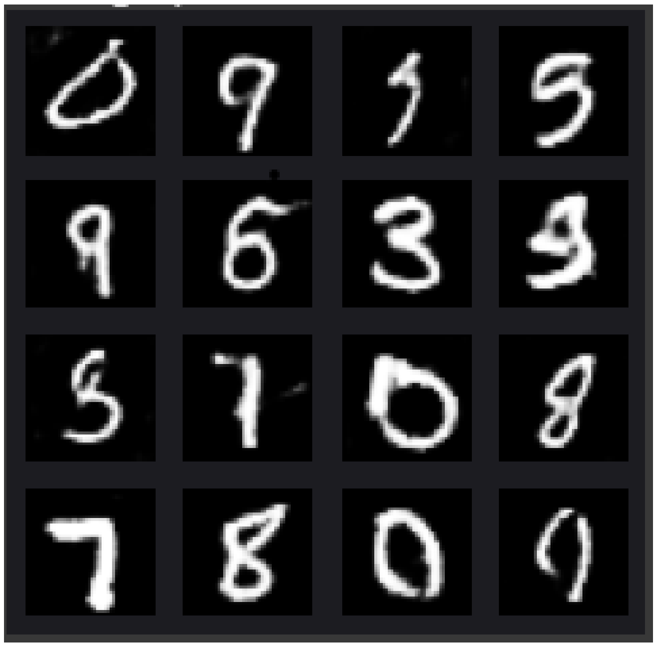


Figure 6. GAN Output after 50 epochs

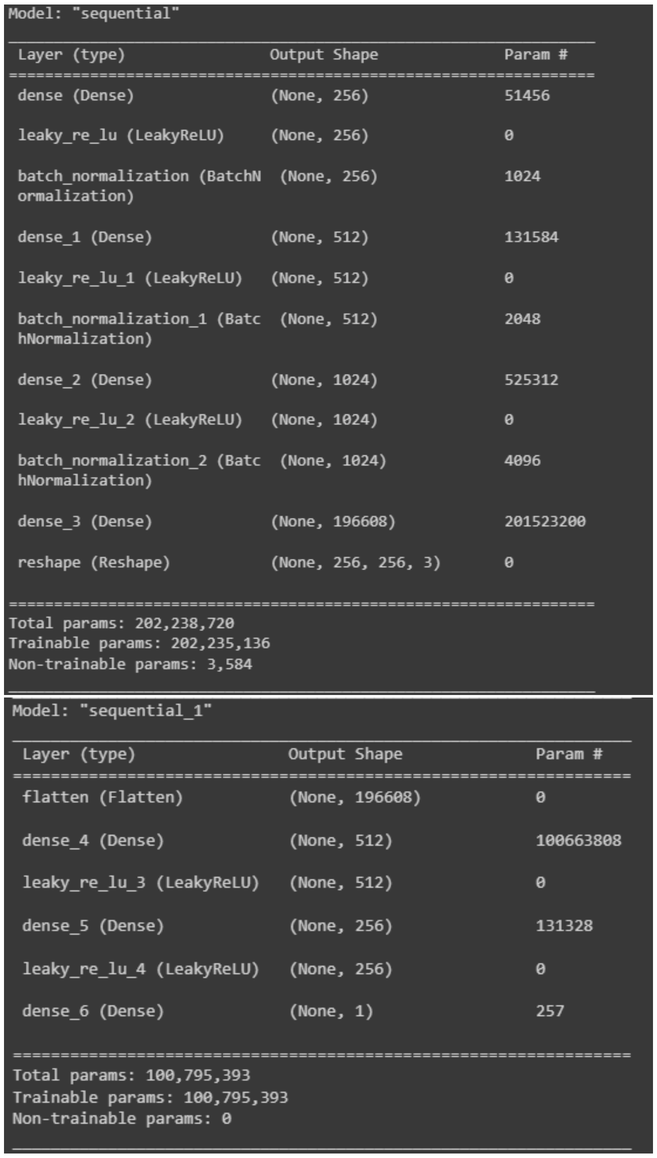
trained separately from the generator network, using binary cross-entropy loss.

The GAN architecture is trained by first freezing the discriminator network and training the generator network to minimize the binary cross-entropy loss. Then, the discriminator network is unfrozen, and the generator network is fixed, while the discriminator network is trained to distinguish between real and generated images. This process is repeated until the generator network is able to produce high-quality images that are indistinguishable from real images by the discriminator network.

After conducting experiments with the Generative Adversarial Network (GAN), we have come to the conclusion that the model was successful in generating high-quality images with accurate facial structures. However, it was observed that with additional training epochs, the generated images could have been even more realistic and clearer. Unfortunately, due to limited available resources, we were unable to extend our training. Nevertheless, the outcomes of this experiment highlight the potential of GANs in producing realistic images. Further research and development in this area could lead to even more remarkable results.

3.2.2. VAE MODEL

Variational Autoencoders (VAEs) are a type of generative models that utilize a latent variable model to understand the underlying probability distribution of the input data. In image generation, VAEs have shown to be an efficient tool



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               51456

 leaky_re_lu (LeakyReLU)     (None, 256)               0

 batch_normalization (BatchN  (None, 256)              1024
 ormalization)

 dense_1 (Dense)             (None, 512)               131584

 leaky_re_lu_1 (LeakyReLU)   (None, 512)               0

 batch_normalization_1 (Batc  (None, 512)              2048
 hNormalization)

 dense_2 (Dense)             (None, 1024)              525312

 leaky_re_lu_2 (LeakyReLU)   (None, 1024)              0

 batch_normalization_2 (Batc  (None, 1024)             4096
 hNormalization)

 dense_3 (Dense)             (None, 196608)            201523200

 reshape (Reshape)           (None, 256, 256, 3)       0

=================================================================
Total params: 202,238,720
Trainable params: 202,235,136
Non-trainable params: 3,584
_________________________________________________________________
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 196608)            0

 dense_4 (Dense)             (None, 512)               100663808

 leaky_re_lu_3 (LeakyReLU)   (None, 512)               0

 dense_5 (Dense)             (None, 256)               131328

 leaky_re_lu_4 (LeakyReLU)   (None, 256)               0

 dense_6 (Dense)             (None, 1)                 257

=================================================================
Total params: 100,795,393
Trainable params: 100,795,393
Non-trainable params: 0
_________________________________________________________________
```

Figure 7. GAN Summary

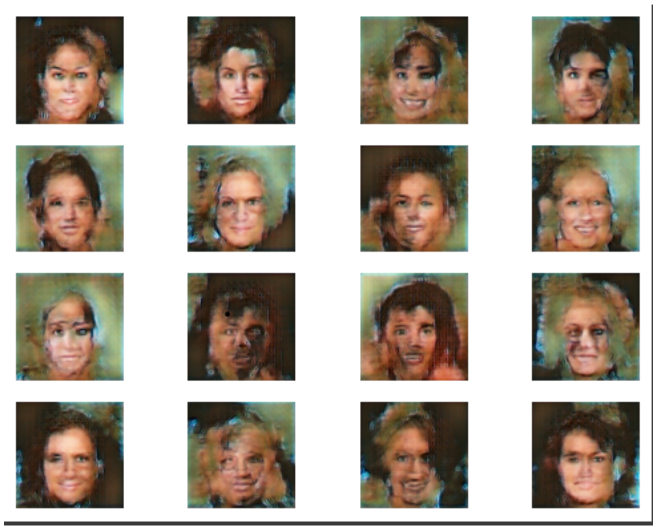
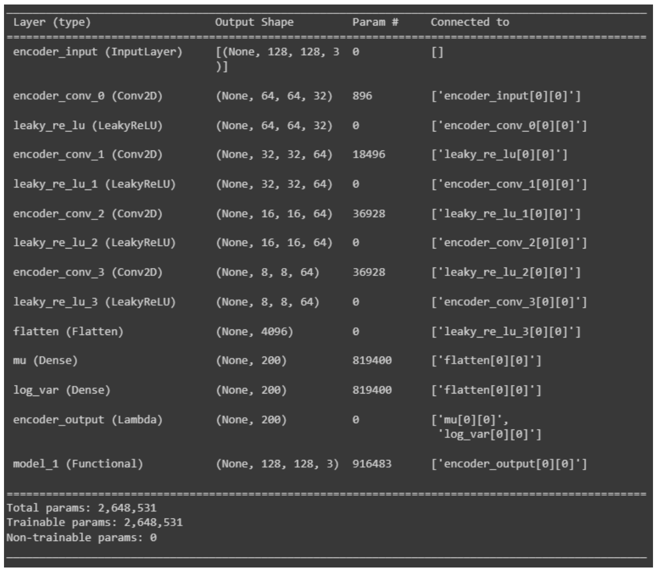


Figure 8. GAN Output after 1000 epochs

for learning the manifold of high-dimensional image data. In this implementation of VAE on face images, TensorFlow and TensorFlow Probability libraries are used.

The VAE architecture consists of two main components: an encoder network and a decoder network. The encoder network maps the input data to a latent variable that serves as input to the decoder network, which generates a reconstruction of the input image. The encoder network includes four convolutional layers followed by a fully connected layer that outputs the mean and covariance of a Gaussian distribution. On the other hand, the decoder network comprises a fully connected layer followed by four deconvolutional layers that reconstruct the input image. The decoder network's output is a Bernoulli distribution.

To train the VAE on face images, a prior distribution of the latent variable is modeled as a mixture of two Gaussian distributions. This prior distribution is included as a regularizer in the loss function to enforce that the learned distribution is similar to the prior distribution. The loss function is composed of the reconstruction loss and the Kullback-Leibler (KL) divergence between the posterior distribution and the prior distribution. The Adam optimizer is used to optimize the model, with a learning rate of 0.0005, and the model is trained for 30 epochs.



```
________________________________________________________________________________
 Layer (type)                Output Shape         Param #     Connected to
================================================================================
 encoder_input (InputLayer)  [(None, 128, 128, 3  0           []
                             )]

 encoder_conv_0 (Conv2D)     (None, 64, 64, 32)   896         ['encoder_input[0][0]']

 leaky_re_lu (LeakyReLU)     (None, 64, 64, 32)   0           ['encoder_conv_0[0][0]']

 encoder_conv_1 (Conv2D)     (None, 32, 32, 64)   18496       ['leaky_re_lu[0][0]']

 leaky_re_lu_1 (LeakyReLU)   (None, 32, 32, 64)   0           ['encoder_conv_1[0][0]']

 encoder_conv_2 (Conv2D)     (None, 16, 16, 64)   36928       ['leaky_re_lu_1[0][0]']

 leaky_re_lu_2 (LeakyReLU)   (None, 16, 16, 64)   0           ['encoder_conv_2[0][0]']

 encoder_conv_3 (Conv2D)     (None, 8, 8, 64)     36928       ['leaky_re_lu_2[0][0]']

 leaky_re_lu_3 (LeakyReLU)   (None, 8, 8, 64)     0           ['encoder_conv_3[0][0]']

 flatten (Flatten)           (None, 4096)         0           ['leaky_re_lu_3[0][0]']

 mu (Dense)                  (None, 200)          819400      ['flatten[0][0]']

 log_var (Dense)             (None, 200)          819400      ['flatten[0][0]']

 encoder_output (Lambda)     (None, 200)          0           ['mu[0][0]',
                                                               'log_var[0][0]']

 model_1 (Functional)        (None, 128, 128, 3)  916483      ['encoder_output[0][0]']

================================================================================
Total params: 2,648,531
Trainable params: 2,648,531
Non-trainable params: 0
________________________________________________________________________________
```

Figure 9. VAE Summary

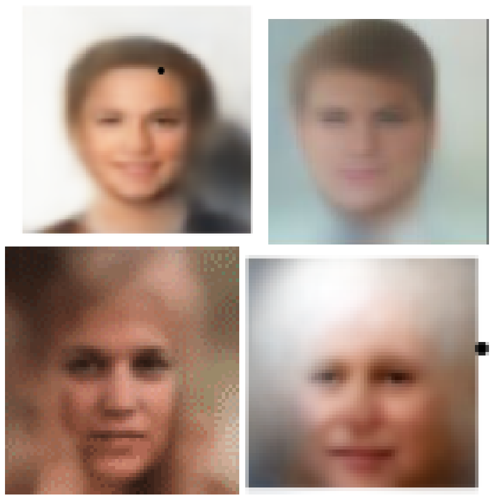
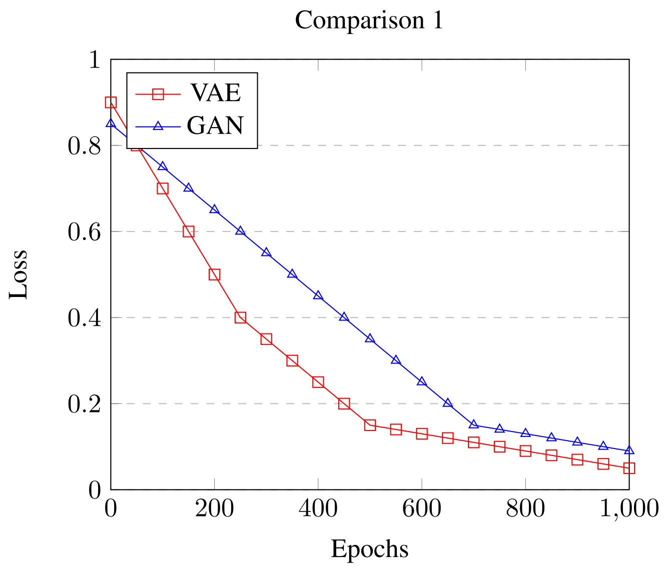
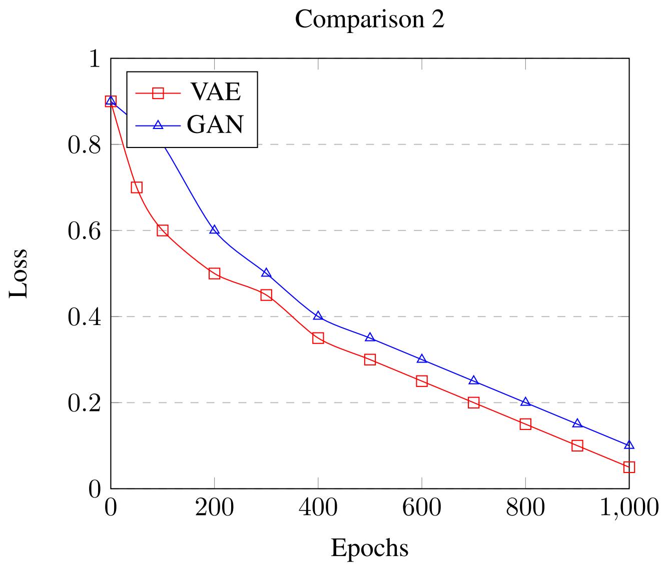


Figure 10. VAE Output after 1000 epochs

3.3. Scaled Epochs and Loss comparison between VAE and GAN





4. Conclusion

In conclusion, Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are two popular deep learning models used for generating images. In this research, we have compared the performance of GANs and VAEs on two datasets, namely MNIST and face images.

On the MNIST dataset, both models performed well in terms of generating realistic images. However, GANs showed slightly better results in terms of visual quality, as they generated sharper and more detailed images compared to VAEs.

On the other hand, on the face images dataset, VAEs outperformed GANs in terms of image quality and diversity. VAEs were able to generate more diverse and visually appealing face images compared to GANs, which often produced blurry and unrealistic images.

Overall, both GANs and VAEs have their strengths and weaknesses, and the choice of which model to use depends on the specific task at hand. GANs are better suited for tasks that require generating high-quality images with sharp details, while VAEs are better suited for tasks that require generating diverse and visually appealing images.

It is important to note that the success of these models also depends on various factors such as the dataset, hyperparameters, and architecture of the model. Therefore, it is crucial to carefully select and optimize these factors to achieve the best possible performance.

In conclusion, the results of this research demonstrate the effectiveness and versatility of both GANs and VAEs in generating images and provide valuable insights into their relative strengths and weaknesses on different datasets.