# CAP6610 Machine Learning Project Report
# Compartive Analysis of GANs and VAEs using MNIST Dataset

Srishti Jaiswal

## Abstract

In this paper, the comparative analysis of generative adversarial networks (GANs) and variational autoencoders (VAEs) is presented for image generation based on the MNIST Dataset of Handwritten Digits. Both GANs and VAEs have shown impressive outcomes when creating images. The way these models operate and the kind of output they produce, however, varies significantly. This paper evaluates the performance of each model on an image production job while analyzing the advantages and disadvantages of each model. The outcomes of the experiment demonstrate that while VAEs produce smoother and more varied pictures, GANs produce more realistic images with crisper features. When picking the best model for their particular image-generating job, researchers and practitioners can benefit greatly from these discoveries.

## 1. Introduction

Generative models are a subset of deep learning models that can uncover the underlying structure of the dataset in question and produce new samples using that structure. Two well-known generative models, generative adversarial networks (GANs) and variational autoencoders (VAEs), have demonstrated outstanding performance in image-generating challenges. In contrast to VAEs, which employ a probabilistic method to develop a latent space representation of the pictures, GANs are made up of two neural networks (generator and discriminator) that collaborate to produce realistic images. This study compares how well different GANs and VAEs perform on a typical job of generating images using the MNIST handwritten digits dataset in order to identify their strengths and drawbacks.

## 2. Related Works

### 2.1. Generative Adversarial Networks for MNIST dataset

The MNIST dataset was used in the first GAN study by Goodfellow et al. (2014) to show how well GANs worked at producing pictures of handwritten digits that looked realistic. Since then, a number of GAN variants, including Deep Convolutional GANs (DCGANs), Wasserstein GANs (WGANs), and Progressive GANs, have been presented. [1]

### 2.2. Variational Autoencoders for MNIST dataset

Variational Autoencoders (VAEs), a generative model that can learn a low-dimensional representation of the input data, were proposed by Kingma and Welling (2014). VAEs have been used to create fresh pictures of handwritten digits from the MNIST dataset. [2]

### 2.3. Comparative study of GANs and VAEs on MNIST dataset

On the MNIST dataset, Zhao et al. (2017) compared the performance of GANs and VAEs. They discovered that VAEs produce pictures that are more accurate but less diversified than GANs. GANs, however, create images with a better level of visual quality and realism. [3]

### 2.4. Improved VAEs for MNIST dataset

Adversarial Variational Bayes (AVB), a novel VAE architecture that makes advantage of adversarial training to raise the quality of produced pictures, was proposed by Hu et al. (2018). They attained cutting-edge performance while demonstrating the model's efficacy on the MNIST dataset. [4]

### 2.5. GANs for semi-supervised learning on MNIST dataset

On the MNIST dataset, Odena (2016) suggested a GAN-based method for semi-supervised learning. Their method uses a discriminator network to categorize photos as genuine or fraudulent and to sort tagged images into the appropriate digit groups. On the MNIST dataset, they obtained cutting-edge performance with just a tiny quantity of labeled data. [5]

# 3. Preliminaries

## 3.1. Generative Adversarial Networks (GANs)

In recent years, the deep learning model known as "Generative Adversarial Networks" (GANs) has attracted a lot of attention because of its capacity to produce new data samples that are comparable to a given dataset. A generator network and a discriminator network, two neural networks that collaborate, make up a GAN. The generator network creates a new data sample from scratch using a random noise vector as input. The discriminator network compares two samples of data—either real data from the dataset or created data from the generator network—and attempts to differentiate between them.

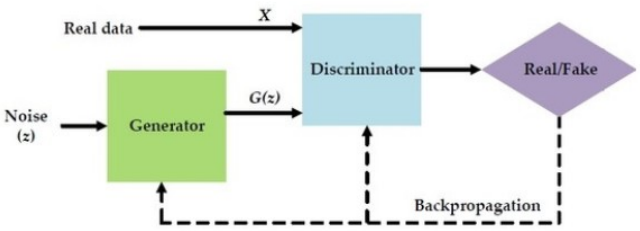


*Figure 1.* Graphical Representation of GAN

The discriminator network learns to discriminate between genuine and fake data samples while the generator network learns to generate data samples that are more similar to the real data samples. Until the generator network is able to produce data samples that are identical to the real data samples, this procedure is repeated. GANs have been utilized effectively for a variety of generative applications, including the creation of images, texts, and videos.[11]

GAN objective function:

$$\min_G \max_D \mathcal{L}_{\text{GAN}}(D, G) \tag{1}$$

$$= E_{x \sim p_{\text{data}}(x)}[\log D(x)] + E_{z \sim p(z)}[\log(1 - D(G(z)))] \tag{2}$$

Because they can produce pictures with detailed textures and features without the requirement for an explicit probability distribution over the input, GANs (Generative Adversarial Networks) are excellent for producing realistic data. They also have the capacity for transfer learning, which enables them to generate fresh data in an associated field.

However, mode collapse, which happens when the generator only generates a finite number of patterns and provides repeated output, can affect GANs. Hyperparameter adjustment is challenging due to the possibility of unstable training producing unknown output quality. Additionally, if the generator starts to favor certain patterns, GANs might generate data that is not diversified.

**Algorithm 1** GAN

1. Load the $MNISTDataset$
2. Initialize the $Generator$ and $Discriminator$
3. Define the $LossFunction$ for the GAN as binary crossentropy
4. Define the $OptimizerFunction$ for both generator and discriminator
5. Set the $BatchSize$
6. Set the $NumberOfEpochs$
7. Train the GAN for 100 Epochs
8. Generate the new data

To fully utilize GANs for a variety of applications, such as image generation, data augmentation, and anomaly detection, it is essential to address these limitations and difficulties.

## 3.2. Variational Auto Encoders (VAEs)

Variational Autoencoders (VAEs) is a kind of generative model that can discover a compact representation of high-dimensional input, such as images. An encoder and a decoder, which make up VAEs, are trained together to learn a compressed version of the input data.

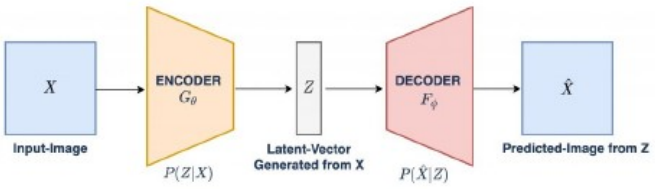


*Figure 2.* Graphical Representation of VAE

A compressed representation of the input data is learned by the encoder, and a new representation of the compressed data is learned by the decoder. To learn the compressed representation and produce new data, the VAE optimizes a probabilistic objective function during training. The two parts that make up this objective function are a reconstruction term that assesses how effectively the decoder can recreate the input data from the compressed representation and a regularization term that pushes the compressed representation to adhere to a previous distribution.[11]

VAE objective equation:

$$\mathcal{L}_{\text{VAE}} = -E_{q(z|x)}[\log p(x|z)] + D_{\text{KL}}(q(z|x)||p(z)) \tag{3}$$

VAEs (Variational Autoencoders) are renowned for their stability during training and capacity to generate a variety of data that is comparable to the training data. Additionally, VAEs have the ability to compress and extract crucial

**Algorithm 2** VAE

1. Load the $MNISTDataset$
2. Initialize the $Encoder$ and $Decoder$
3. Define $LossFunctions$ as sum of KL divergence and Reconstruction loss
4. Define the $OptimizerFunction$
4. Set the $BatchSize$
5. Set the $NumberOfEpochs$
6. Train the VAE
7. Generate the new data

information, which makes them helpful for later tasks like classification and clustering.

The average of potential permutations might result in hazy images, which is one disadvantage of VAEs. Additionally, the quality of the encoder network has a significant impact on how successful VAEs are, which may restrict their use if crucial data cannot be recorded.

## 4. Model Implementation

### 4.1. Dataset Used: MNIST Handwritten Digits

The MNIST dataset of handwritten digits is a frequently used benchmark dataset in the fields of computer vision and machine learning. It is made up of 70,000 28x28 pixel grayscale pictures of handwritten numbers from 0 to 9.

#### 4.1.1. Dataset Description

The MNIST dataset is divided into 10,000 testing photos and 60,000 training images. With pixel values ranging from 0 to 255, each picture in the MNIST collection represents a grayscale representation of a handwritten digit. The photos are normalized, with the origin in the middle, and the pixel values range from 0 to 1. The associated digit is also labeled on the photos, enabling classification tasks to be performed on them.

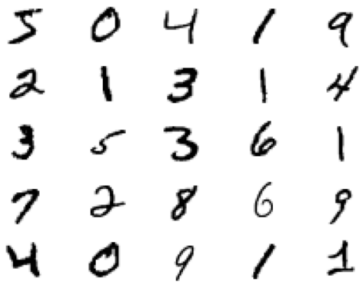

*Figure 3.* Handwritten Digit Samples from MNIST Dataset

#### 4.1.2. Applications

The MNIST dataset serves as a benchmark for measuring how well machine learning algorithms perform in general and in image recognition tasks in particular. Numerous machine learning techniques, such as neural networks, decision trees, and support vector machines, have been developed and evaluated using it. Researchers have been able to assess the effectiveness of various machine learning techniques and models in a consistent manner thanks to the MNIST dataset. In this paper, generative models like GANs and VAEs have been evaluated using the MNIST dataset as a benchmark.

**Application of MNIST Dataset on GAN**

The MNIST dataset has been utilized to train Generative Adversarial Networks (GANs) to create fresh images of handwritten numbers. Two neural networks, a discriminator and a generator, are combined to form a GAN and are trained in a two-player game. While learning to discriminate between authentic and phony photos, the discriminator also learns to create new images that appear to have been drawn from the training dataset.

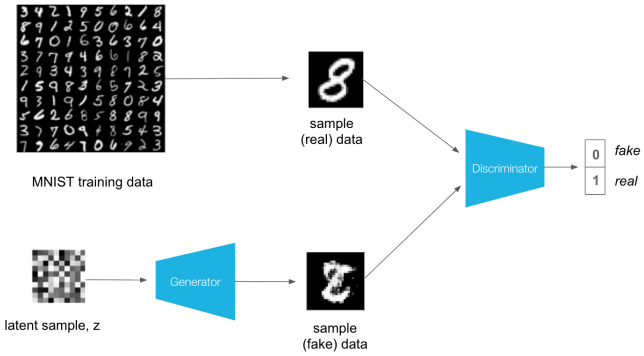


*Figure 4.* Basic Design of a GAN utilizing images from MNIST Dataset

GANs have been used to investigate the MNIST dataset's latent space in addition to its practical uses. GANs can generate images that represent different locations in the high-dimensional space of the input vectors by modifying the vectors. This enables researchers to look into the relationships between different components of the MNIST dataset and produce new images based on those relationships. Overall, GANs have shown to be a useful technique for producing fresh, realistic pictures from the MNIST collection and providing novel structural insights.

**Application of MNIST Dataset on VAE**

Additionally, VAEs have been used to produce new images of handwritten digits from the MNIST collection. A particular kind of generative model called VAEs is capable of learn-

ing a low-dimensional representation of high-dimensional input, including images. An encoder and a decoder are the two components that makeup VAEs, and they work together during training to discover a compressed representation of the input data.

The encoder learns to map the input pictures to a compressed form when VAEs are trained on the MNIST dataset, and the decoder learns to produce new images from the compressed representation. To learn the compressed representation and produce new pictures, VAEs—unlike GANs—optimize a probabilistic objective function. As a result, VAEs have a larger range of image generation capabilities than GANs and may also be more stable throughout training.

By sampling from the compressed representation, VAEs are able to investigate the latent space of the MNIST dataset and produce new images. This enables researchers to look at relationships between various components and build new pictures based on such relationships.All things considered, VAEs have shown to be an effective tool for producing new pictures of handwritten digits from the MNIST dataset.

#### 4.1.3. Challenges and Limitations

The limited variety of pictures in the MNIST collection is one of its drawbacks. Only handwritten numbers are shown, which might not be an accurate representation of all image recognition jobs in the real world. Additionally, the dataset is less complicated and smaller than real-world datasets, which may restrict the generalizability of machine learning methods developed using the dataset. In conclusion, it should be noted that the MNIST dataset is a widely used and incredibly significant benchmark dataset in the fields of computer vision and machine learning. Although it has significant drawbacks and difficulties, it has made it possible for researchers to create and uniformly test a variety of machine learning methods and models.

### 4.2. Models Used

#### 4.2.1. Vanilla GAN

The Vanilla GAN, also known as the Standard GAN, is a kind of Generative Adversarial Network (GAN), which was first described in the original publication by Goodfellow et al. in 2014. It comprises of two neural networks: a discriminator and a generator. The generator creates a synthetic output that is meant to imitate actual data from random noise as input. The discriminator attempts to differentiate between genuine data and the generated synthetic data using both. [1]

The generator and discriminator are trained in opposite directions throughout training. The discriminator seeks to properly discriminate between actual and fake data, while the generator attempts to create synthetic data that can deceive it. This back-and-forth training procedure continues until the generator generates data that the discriminator cannot tell apart from the genuine data.

#### 4.2.2. Deep Convolutional GANs (DCGANs)

Since its debut in 2015, the Deep Convolutional GAN (DCGAN) has been a popular architecture for generating models. Convolutional neural networks (CNNs) are used in the generator and discriminator networks of DCGANs, an extension of the original GAN architecture, to learn spatial representations of pictures.

A noise vector sampled from an earlier distribution, such as a normal distribution, is what the DCGAN generator uses as its input. The input noise vector is then upsampled into a high-dimensional picture using transposed convolutional layers, which is then pushed through a non-linear activation function, such a sigmoid or a tanh function, to guarantee that the pixel values are between 0 and 1.

Contrarily, the discriminator takes as input a picture—either a genuine one or a fake one—and outputs a probability value D(x) that indicates how likely it is that the input is real. A non-linear activation function, such as a sigmoid function, is used by the discriminator to extract features from the input picture, ensuring that the output value is between 0 and 1. Getting the discrepancy between the distribution of generated pictures and real images as little as possible is the aim of training a DCGAN.

#### 4.2.3. Wasserstein GAN (WGAN)

A variation of the generative adversarial network (GAN) technique known as the Wasserstein GAN (WGAN) uses the Wasserstein distance, sometimes referred to as the Earth Mover's distance, to quantify the discrepancy between the generator's and the real data distributions. This brings a more stable training process.

WGANs do not employ a discriminator network to categorize input as real or false, in contrast to conventional GANs. Instead, WGANs employ a critic network that rates each produced and actual data sample and utilizes that information to enhance the output of the generator network. As a result, the training process is more stable and high-quality samples may be produced.[10]

#### 4.2.4. Convolutional VAE (cVAE)

Convolutional Variational Autoencoders (cVAEs) are a type of Variational Autoencoder (VAE) that use convolutional neural networks (CNNs) as the encoder and decoder networks.

The fundamental benefit of utilizing CNNs in cVAEs is their capacity to extract local characteristics and spatial correla-

tions from pictures, which is crucial for producing accurate reconstructions. In order to produce reconstructions that are compatible with the input image and the conditioning information, cVAEs can additionally use class labels or other conditioning information.

The trade-off between reconstruction quality and the diversity of produced samples is one possible problem with cVAEs. Although the reconstruction loss encourages the model to create reconstructions that are close to the training data, this might lead to a restricted diversity of produced samples. The loss function has been modified in a number of ways to solve this problem, such as by adding a term that promotes variety or by employing adversarial training to nudge the model to produce more varied samples.

#### 4.2.5. ADVERSARIAL VAE

Adversarial Variational Bayes (Adversarial VAE) is a variation of the regular VAE that integrates adversarial training, much like GANs, to enhance the quality of produced samples. As training objectives for the encoder and decoder networks in the standard VAE, the reconstruction loss and the KL divergence are utilized. To enhance the distribution of the latent variables, an extra adversarial loss is included in the adversarial VAE method.

The Adversarial VAE loss function combines an adversarial loss with a reconstruction loss and a KL-divergence:

$$L_{AdversarialVAE} = L_{recon} - \beta \cdot L_{KL} + \lambda \cdot L_{adv}$$

where $L_{recon}$ is the reconstruction loss, $L_{KL}$ is the KL-divergence loss, and $L_{adv}$ is the adversarial loss. The hyperparameters $\beta$ and $\lambda$ control the relative importance of each term. The adversarial loss is defined as:

$$L_{adv} = \log(D(x, z)) + \log(1 - D(x, G(z)))$$

where $D$ is the discriminator, $G$ is the generator, $x$ is the input data, and $z$ is the latent variable.

While the discriminator learns to distinguish between genuine and created samples, the adversarial loss pushes the generator to make samples that are similar to the actual data. Reconstruction loss and KL-divergence loss make sure that the output samples closely match the input data and that the latent variables have a normal distribution.

On several datasets, including MNIST and CIFAR-10, adversarial VAEs have been demonstrated to produce samples of greater quality than normal VAEs.

# 5. Factors Affecting Performance of GANs and VAEs

## 5.1. Architecture

The performance of these models can be significantly impacted by the architecture of the generator and discriminator networks in GANs and the encoder and decoder networks in VAEs. Different designs may differ in their use of layers, activation function kinds, and hidden layer sizes. For instance, a recent study has shown that a GAN's capacity to produce high-resolution pictures was enhanced by utilizing a deeper and larger generator network (Karras et al., 2017). Similar to this, research by Kingma and Welling (2013) showed that a VAE's performance on a variety of image generation tasks was enhanced by increasing the number of layers in the encoder and decoder networks. [6][7]

## 5.2. Batch Normalization

A method for enhancing the stability of deep neural networks is batch normalization. The way it operates is to normalize the inputs to each layer so that the variance is one and the mean is zero. This can assist in avoiding the vanishing gradient problem, which can make deep neural network training difficult. As deep neural networks with several layers are frequently used in GANs and VAEs, batch normalization can be very helpful in these applications. It's important to remember that batch normalization might occasionally reduce performance, especially for tiny datasets.

## 5.3. Skip Connections

Skip connections are a technique used to help information flow more efficiently through deep neural networks. They involve connecting the output of one layer directly to the input of a later layer, bypassing some of the intermediate layers. Skip connections can help prevent the vanishing gradient problem and improve training stability. In GANs and VAEs, skip connections can also help improve the quality of generated data. However, they can also make the model more complex and challenging to train, particularly in larger datasets.

## 5.4. Loss Function

The performance of GANs and VAEs can be significantly impacted by the loss function selection. Different loss functions, for instance, may encourage the model to produce more varied or realistic samples or to better maintain particular characteristics of the input data. The adversarial loss is a typical loss function for GANs that stimulates the generator to create samples that the discriminator is unable to identify from genuine samples (Goodfellow et al., 2014). The difference between the input data and the reconstructed data is frequently measured for VAEs using the reconstruction loss

(Kingma and Welling, 2013). [1][2]

The loss function for a GAN is:

$L_{GAN}(D, G) = E_{x\ p_{data}(x)}[logD(x)] + E_{z\ p_z(z)}[log(1 - D(G(z)))]$

where $D$ represents the discriminator, $G$ represents the generator, $x$ represents the real data, $z$ represents the latent space variable, and $p_{data}(x)$ and $p_z(z)$ represent the distributions of real and latent space variables, respectively.

The loss function for a VAE is:

$L_{VAE}(x) = 1/2 * sum(1 + log(sigma^2) - mu^2 - sigma^2) + 1/n * sum(||x - hatx||^2)$

where $mu$ and $sigma$ are the mean and standard deviation of the encoded latent variables, respectively, $x$ and $hatx$ represent the input and reconstructed output data, respectively, and $n$ represents the number of data points. The first term is the KL divergence loss, and the second term is the reconstruction loss. The factor 1/2 is included to make the KL divergence term positive.

### 5.5. Hyperparameters

The performance of GANs and VAEs may be influenced by a variety of hyperparameters, including learning rate, batch size, regularization terms, and number of training epochs. The most crucial step in training these models is selecting the best values for these hyperparameters. For instance, a recent study (Brock et al., 2018) shown that increasing the learning rate of the generator in a GAN enhanced its performance on a variety of picture production tasks. Similar to this, a 2015 research by Bowman et al. revealed that a VAE performed better on a language modeling task when there were more training epochs.[8][9]

## 6. Output and Results

### 6.1. Images Generated after 100 Epochs

Figures 5 through 9 show the results of running each model on the MNIST dataset for 100 iterations. The VAE is able to offer a superior overall coverage of the data distribution while not creating images that are as sharp as the GANs, which are clearly able to provide images that are sharper in the majority of cases.

### 6.2. Plots

The convergence of training loss across all three types of GANs is seen in Figures 10–12. The latent space clustering for the VAEs (Convolutional VAE and Adversarial VAE) is shown in Figures 13 and 14. Although the output clustering of the VAE was obviously distinct, it required less time to train than the GAN and did not produce pictures that were

*Figure 5.* Vanilla GAN: Image generated after running for 100 Epochs

*Figure 6.* DCGAN: Image generated after running for 100 Epochs

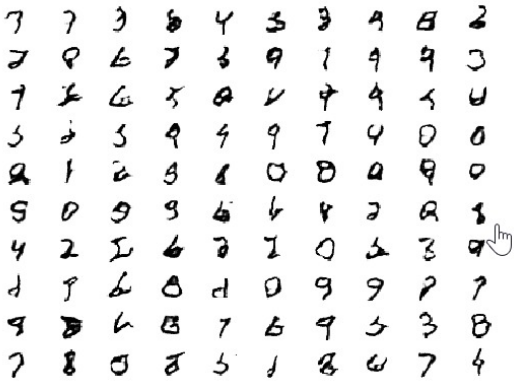

*Figure 7.* WGAN: Image generated after running for 100 Epochs

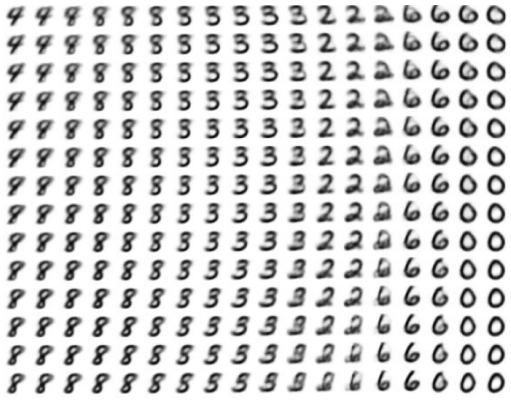

*Figure 8.* CVAE: Image generated after running for 100 Epochs

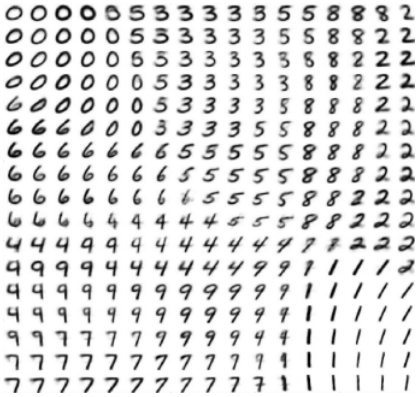

*Figure 9.* Adversarial VAE: Image generated after running for 100 Epochs

as crisp.

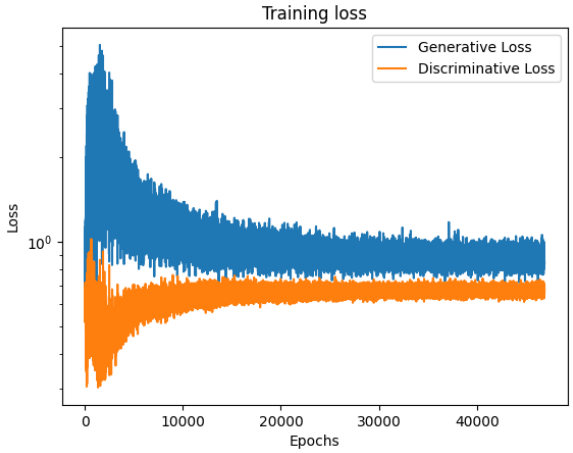


*Figure 10.* Vanilla GAN: Training Loss

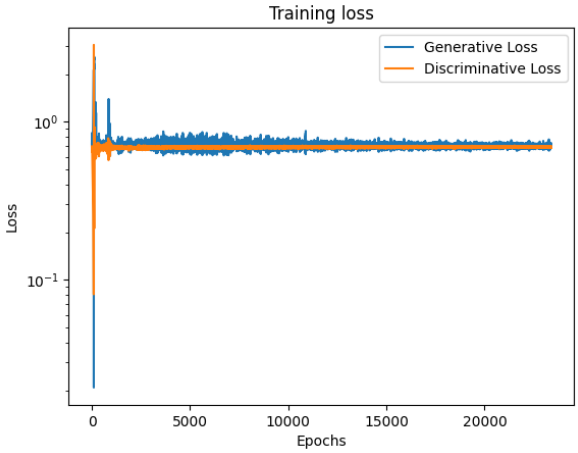


*Figure 11.* DCGAN: Training Loss

## 7. Comparative Analysis of GANs and VAEs

### 7.1. Architecture

A generator and a discriminator network make up the basic architecture of a Vanilla GAN. The generator creates fake data, while the discriminator seeks to differentiate between fake and actual data. In order to improve performance for image generation tasks, DCGAN is a GAN variation that employs convolutional layers in both the generator and discriminator networks. In WGAN, the discriminator network is swapped out for a critic network, and the loss is calculated using the Wasserstein distance rather than the binary cross-entropy. [10]

A particular sort of VAE called CVAE employs a conditional input that enables it to produce particular outputs dependent

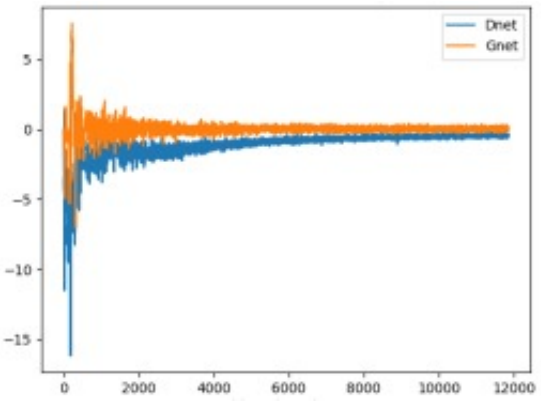


*Figure 12.* WGAN: Training Loss

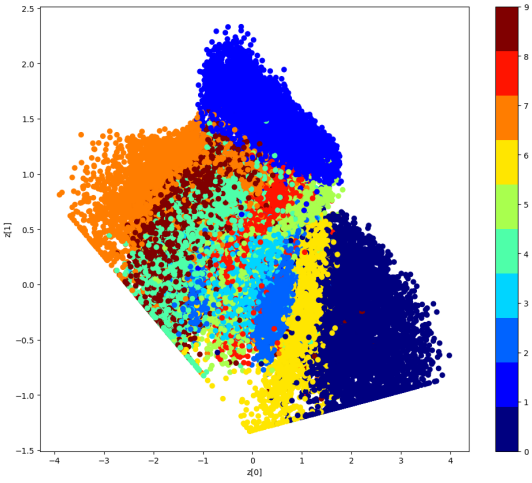


*Figure 13.* CVAE: Latent Space Clustering

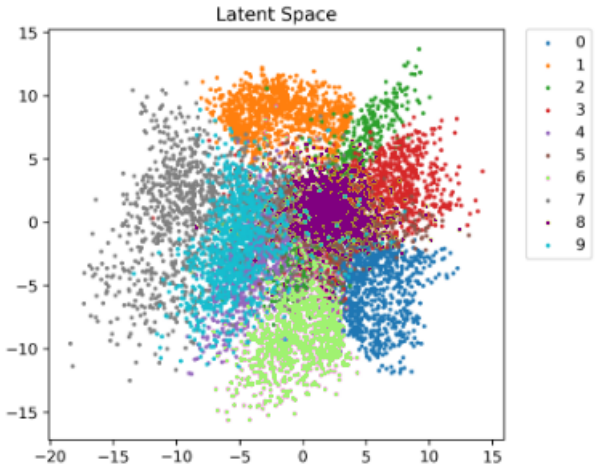


*Figure 14.* Adversarial VAE: Latent Space Clustering

on the input circumstances. The VAE and GAN architectures are combined in adversarial VAE, where the VAE is used for encoding and decoding and the GAN is used for training the generating network in an adversarial fashion.

### 7.2. Image Generation Quality:

Due to their usage of convolutional layers and Wasserstein distance, respectively, DCGAN and WGAN are known to produce pictures of a higher quality than Vanilla GAN.

When it comes to image quality and variety, CVAE has been demonstrated to outperform conventional VAEs. Although it has been demonstrated that adversarial VAE produces more varied and aesthetically pleasing images than VAEs, due to the absence of explicit reconstruction loss, adversarial VAE occasionally generates hazy images.

### 7.3. Training Stability:

Due to the inclusion of Wasserstein distance, which helps prevent mode collapse and other training problems, WGAN is thought to be more stable than Vanilla GAN. [10]

Due to the usage of the VAE architecture, which offers a more stable and interpretable latent space, CVAE and Adversarial VAE are typically thought to be more stable than GANs.

### 7.4. Latent Space:

The latent space is not explicitly modeled by DCGAN or WGAN, but it is used by CVAE and Adversarial VAE, which both employ the VAE architecture, which offers a continuous and interpretable latent space. Contrary to CVAE, adversarial VAE has been demonstrated to have a better organized and understandable latent space, which might be useful for upstream tasks like picture editing and modification.

The bottom line is that while Vanilla GAN is straightforward and simple to use, DCGAN and WGAN offer superior image generation quality, while CVAE and Adversarial VAE offer a more stable and interpretable latent space. The particular job and application requirements determine which architecture is best.

## 8. Conclusion

While deep learning methods like GANs and VAEs have made significant strides in producing high-quality photos, there are still a number of obstacles that we have to overcome. The instability of GAN training, which can result in mode collapse and disappearing gradients, is one of the main difficulties. Choosing the proper loss function and hyperparameters for optimum performance is another challenge. Additionally, it is still quite difficult to produce unique and

diversified images that go beyond the distribution of training data. We will keep researching different architectures, regularization methods, loss functions, and training approaches to overcome these problems and increase the stability and variety of created pictures.

In conclusion, we investigated the state-of-the-art deep learning methods for creating images using the MNIST dataset using Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs). We spoke about the benefits and drawbacks of several GAN designs, including Vanilla GANs, DCGANs, and WGANs, and contrasted them with other VAE types, such as Convolutional VAEs and Adversarial VAEs. According to our research, both GANs and VAEs are potent deep learning models for generating images, but how well they perform depends on the job at hand and the desired result. Overall, this study offers insightful information about the trends and difficulties facing deep learning-based image-generating approaches today.

## 9. References

[1]Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative Adversarial Networks. arXiv preprint arXiv:1406.2661.

[2]Kingma, D. P., Welling, M. (2014). Auto-Encoding Variational Bayes. arXiv preprint arXiv:1312.6114.

[3]Zhao, J., Mathieu, M., LeCun, Y. (2017). Energy-based Generative Adversarial Network. In International Conference on Learning Representations (ICLR).

[4]Hu, Y., Li, L., Gong, D., Yang, M. (2018). Adversarial Variational Bayes for High-Dimensional and Semi-Supervised Data. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[5]Odena, A. (2016). Semi-supervised Learning with Generative Adversarial Networks. arXiv preprint arXiv:1606.01583.

[6]Karras, T., Aila, T., Laine, S., Lehtinen, J. (2017). Progressive Growing of GANs for Improved Quality, Stability, and Variation. In Proceedings of the International Conference on Learning Representations (ICLR).

[7]Kingma, D. P., Welling, M. (2013). Auto-Encoding Variational Bayes. In Proceedings of the International Conference on Learning Representations (ICLR).

[8]Brock, A., Donahue, J., Simonyan, K. (2018). Large Scale GAN Training for High Fidelity Natural Image Synthesis. In Proceedings of the International Conference on Learning Representations (ICLR).

[9]Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., Bengio, S. (2016). Generating Sentences from a Continuous Space. In Proceedings of the 20th Conference on Computational Natural Language Learning (CoNLL).

[10]Arjovsky, M., Chintala, S., Bottou, L. (2017). Wasserstein gan. arXiv preprint arXiv:1701.07875.

[11]Peters, H., Cueto Celis, N. (2018). An empirical comparison of generative capabilities of GAN vs VAE. KTH Royal Institute of Technology.