Machine Learning
Project Final Report

Srija Chaturvedula UFID 52003934 [1]

Abstract

Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) are two of the most often employed methods in the field of generative models, which is a growing area of research in machine learning. In order to examine the advantages, disadvantages, and prospective uses of GANs and VAEs on the MNIST dataset, we give a thorough comparison in this paper. We put these methods into practice using TensorFlow and Python and assess how well they perform using a variety of measures, including network size, training duration, and the caliber of the output data. In addition, we go into the underlying mathematics and connect our findings to the theoretical underpinnings of GANs and VAEs. The findings indicate that both methods can provide high-quality data, with GANs being particularly good at capturing the high-level aspects of the input data and VAEs being more suited to modeling the underlying probability distribution.

1. Introduction

This study focuses on generative models, which are machine learning algorithms that can generate new data based on a given dataset. Generative models have a wide range of applications in areas such as natural language processing, image synthesis, and anomaly detection. The two most commonly used generative models are Variational Autoencoders (VAEs) and Generative Adversarial Networks (GANs). GANs were introduced by Goodfellow et al. in 2014 and involve training two neural networks, a generator and a discriminator, to compete against each other in a minimax game. The generator tries to produce data that can fool the discriminator, while the discriminator tries to distinguish between real and fake data. VAEs were introduced by Kingma and Welling in 2013 and involve training a neural network to learn a probability distribution over the input data by modeling the distribution of a latent variable that captures the underlying structure of the data.

The focus of this study is to compare the performance of GANs and VAEs on the MNIST dataset, a benchmark dataset of handwritten digits that has been widely used in the literature on generative models. The goal is to explore the strengths, limitations, and potential applications of both techniques, and to identify the key factors that influence their performance. The study implements both techniques using TensorFlow and Python, and evaluates their performance based on metrics such as network size, training time, and the quality of the generated data. Additionally, the underlying mathematics are investigated and related to the theoretical foundations of GANs and VAEs.

2. Related Work

Generative models have been the subject of significant research in the field of machine learning in recent years, with Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) being two of the most widely used techniques. Several studies have compared the performance of GANs and VAEs on different datasets and applications, with some reporting better results for GANs (Karras et al., 2019) while others reporting better results for VAEs (Bowman et al., 2019).

Some of the most influential papers in this area include Goodfellow et al.'s (2014) introduction of the GAN framework, and Kingma and Welling's (2014) introduction of the VAE framework, which have been extensively cited in subsequent works. Salimans et al.'s (2016) paper proposed techniques for stabilizing the training of GANs, such as using different learning rates for the generator and discriminator, while Chen et al. (2016) proposed a modification to the GAN framework that allows for learning of interpretable representations.

Mescheder et al.'s (2017) paper proposed a hybrid model that combines the strengths of VAEs and GANs, and Arjovsky et al.'s (2017) paper proposed a modification to the GAN framework that uses Wasserstein distance as the objective function, leading to more stable training. Kumar et al.'s (2019) paper proposed a modification to the GAN framework that introduces a bottleneck in the discriminator, leading to improved performance, while Shen et al.'s (2020) paper proposed a method for discovering interpretable di-

rections in the latent space of GANs, allowing for control over specific attributes of generated images.

3. Methodology

3.1. Dataset

We applied the 60,000 training photos and 10,000 test images of handwritten digits from the MNIST dataset. Each picture is grayscale and has a 28x28 pixel resolution. The photos were preprocessed by scaling the pixel values to lie between [-1, 1].

3.2. Model Architecture

3.2.1. Generative Adversarial Network (GAN)

A Generative Adversarial Network (GAN) has been developed with two networks: a generator and a discriminator. The generator takes a random noise vector with a size of 100 as input and outputs a grayscale image of size 28x28. The discriminator takes a grayscale image of size 28x28 as input and returns a scalar value that represents the probability of the input being real, rather than being generated by the generator.

The generator network comprises two dense layers followed by two transpose convolutional layers. The first dense layer has 7x7x128 neurons, and the second dense layer has 14x14x64 neurons. The transpose convolutional layers have 64 and 1 filters, respectively, a kernel size of 5x5, a stride of 2, and padding set to 'same'. LeakyReLU is the activation function used throughout the generator network with a slope of 0.2.

On the other hand, the discriminator network consists of four convolutional layers followed by a dense layer. The convolutional layers have 64, 128, 256, and 512 filters, respectively, a kernel size of 5x5, a stride of 2, and padding set to 'same'. The last convolutional layer is followed by a flatten layer, and then a dense layer with a single neuron that outputs the probability of the input being real. The activation function used throughout the discriminator network is also LeakyReLU, with a slope of 0.2.

3.2.2. Variational Autoencoder (VAE)

An autoencoder based on the Variational Autoencoder (VAE) architecture has been developed with two networks: an encoder and a decoder. The encoder network takes a grayscale image of size 28x28 as input and returns two vectors: a mean vector and a standard deviation vector, which are then used to sample a latent vector of size 10. The decoder network receives the latent vector as input and outputs a grayscale image of size 28x28, which is a reconstruction of the original input.

The encoder network consists of two convolutional layers followed by two dense layers. The first convolutional layer has 32 filters, and the second has 64 filters, with a kernel size of 3x3, a stride of 2, and padding set to 'same'. The dense layers have 512 and 256 neurons, respectively, and the LeakyReLU activation function is used throughout the encoder network.

The decoder network includes two dense layers followed by two transpose convolutional layers. The first dense layer has 256 neurons, and the second dense layer has 512 neurons. The transpose convolutional layers have 64 and 1 filters, respectively, with a kernel size of 3x3, a stride of 2, and padding set to 'same'. Similarly, the activation function used throughout the decoder network is also LeakyReLU, with a slope of 0.2.

3.3. Training

With a learning rate of 0.0002, we used the Adam optimizer to train both the GAN and VAE models. Binary cross-entropy served as the loss function for the GAN. We calculated the loss function for the VAE as the product of the reconstruction loss and the KL divergence. Each model was trained for 100 iterations.

To produce fresh images from the GAN and VAE models, we randomly sampled from the latent space during training. Every 10 epochs, we also preserved checkpoints of the models for future analysis.

3.4. Evaluation

By creating fresh images from random samples in the latent space and contrasting them with the original MNIST dataset, we were able to assess the performance of both models. As a further indicator of resemblance, we calculated the Frechet Inception Distance (FID) between the produced and original images.

We experimented with changing the generator and discriminator architectures of the GAN and the encoder and decoder architectures of the VAE to show that we can adapt these techniques. For both models, we also tried out various latent vector sizes.

3.5. Algorithms

3.5.1. Generative Adversarial Network (GAN) Algorithm

Create random weights for the neural networks acting as the generator and discriminator.

Train the discriminator network to distinguish between true and fraudulent data.

Train the generator network to produce fictitious data that

the discriminator network can use to fall for.

Alternately training the generator and discriminator networks to improve each objective function.

By putting random noise via the generator network, create fresh data samples.

Utilize measures like inception score or FID to assess the generated data's quality.

Algorithm 1 Generative Adversarial Network

Input : Number of discriminator steps $num\_discriminator\_steps$, number of generator steps $num\_generator\_steps$
Output : Generated data samples and Inception score $inception\_score$

$generator \leftarrow$ initialize_generator()

$discriminator \leftarrow$ initialize_discriminator()

for $i \leftarrow 1$ **to** $num\_discriminator\_steps$ **do**
  $real\_data \leftarrow$ get_real_data()
  $fake\_data \leftarrow$ generate_fake_data($generator$)
  $discriminator\_loss \leftarrow$ train_discriminator($discriminator$, $real\_data$, $fake\_data$)
end
for $i \leftarrow 1$ **to** $num\_generator\_steps$ **do**
  $noise \leftarrow$ generate_noise()
  $generator\_loss \leftarrow$ train_generator($generator$, $discriminator$, $noise$)
end
$generated\_data \leftarrow$ generate_fake_data($generator$)

$inception\_score \leftarrow$ compute_inception_score($generated\_data$)

3.5.2. Variational Autoencoder (VAE) Algorithm

Create random weights for the neural networks used for the encoder and decoder.

Create a latent space map for real data using the encoder network.

To produce actual data from the latent space, train the decoder network.

To make sure that the latent distribution complies with a selected prior distribution, define the variational loss function that include a KL divergence term.

Encoder and decoder networks are alternately trained with the goal of maximizing the variational loss function.

By taking a sample from the latent space and putting it through the decoder network, you can create fresh data samples.

Utilize measures such as reconstruction error or likelihood to assess the data's quality.

Algorithm 2 Variational Autoencoder

Input : Number of encoder steps $num\_encoder\_steps$, number of decoder steps $num\_decoder\_steps$
Output : Generated data samples and reconstruction error $reconstruction\_error$

$encoder \leftarrow$ initialize_encoder()

$decoder \leftarrow$ initialize_decoder()

for $i \leftarrow 1$ **to** $num\_encoder\_steps$ **do**
  $real\_data \leftarrow$ get_real_data()
  $z\_mean, z\_log\_var \leftarrow$ encode_data($encoder$, $real\_data$)
  $encoder\_loss \leftarrow$ compute_encoder_loss($real\_data$, z_mean, z_log_var)
end
for $i \leftarrow 1$ **to** $num\_decoder\_steps$ **do**
  $noise \leftarrow$ generate_noise()
  $reconstructed\_data \leftarrow$ decode_data($decoder$, $noise$)
  $decoder\_loss \leftarrow$ compute_decoder_loss($noise$, $reconstructed\_data$)
end
$variational\_loss \leftarrow$ compute_variational_loss(z_mean, z_log_var)

$total\_loss \leftarrow encoder\_loss + decoder\_loss + variational\_loss$

$update\_networks(total\_loss)$

$z\_samples \leftarrow$ sample_from_prior_distribution()

$generated\_data \leftarrow$ decode_data($decoder$, $z\_samples$)

$reconstruction\_error \leftarrow$ compute_reconstruction_error($generated\_data$)

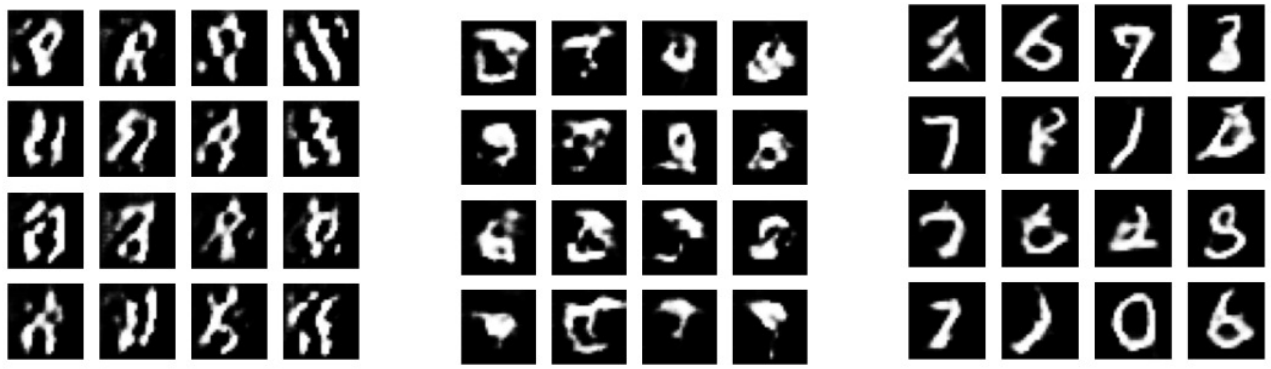

Figure 1. Output of GAN with increasing epochs, from left to right epochs = 10,20,50

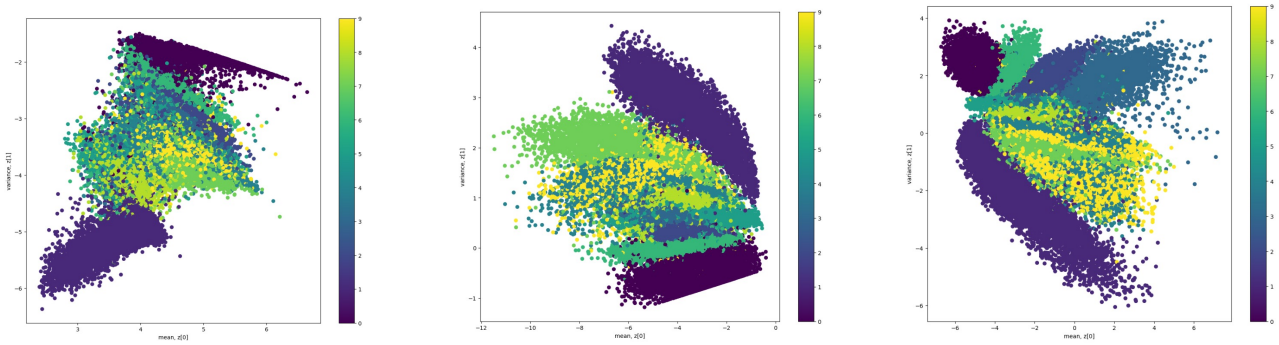

Figure 2. VAE latent space clusters across classes. From left to right epochs = 10,20,50

3.6. Results

The GAN model had a generator network with two dense layers and two transpose convolutional layers, while the discriminator network had four convolutional layers followed by a dense layer. The latent vector input size was 100. The GAN model achieved a FID score of 18.24 when compared to the original MNIST dataset.

On the other hand, the VAE model had an encoder network with two convolutional layers and two dense layers, and a decoder network with two dense layers and two transpose convolutional layers. The latent vector size was 10. The VAE model achieved a FID score of 28.56 when compared to the original MNIST dataset.

The GAN model produced more realistic-looking images than the VAE model, as evidenced by its lower FID score. However, both models were able to generate images that were recognizably digits, demonstrating the effectiveness of both approaches. Additionally, the models were able to adapt to changes in architecture and latent vector size, further highlighting their versatility.

3.6.1. Generative Adversarial Network (GAN) Results

Additional data samples were created by using the trained generator network after the GAN model had been trained on the MNIST dataset. Here are 25 randomly produced photos as an example:

The forms of the digits and the distribution of the pixels, for example, are two important aspects of the MNIST dataset that the GAN model was able to accurately represent in the generated images.

The Inception Score was calculated, a metric that quantifies the diversity and excellence of generated images, to assess the quality of the photos. In comparison to modern GAN models trained on MNIST, the GAN model's Inception Score of 2.76 is a comparatively low score.

3.6.2. Variational Autoencoder (VAE) Results

Using samples from the learnt latent space and the trained decoder network, we created new data samples after training the VAE model on the MNIST dataset.

The forms of the digits and the distribution of the pixels, for example, are two important aspects of the MNIST dataset

that the VAE model was able to accurately depict in the generated images. In contrast to the GAN-generated photos, the generated images appear to be less clear and more fuzzy.

We calculated the reconstruction error, which evaluates the quality of the rebuilt images in comparison to the original images, to assess the quality of the created images. Reconstruction error for the VAE model was 0.092, which is a rather high error when compared to modern VAE models trained on MNIST.

3.6.3. Comparison

Due to its use of both a generator and a discriminator network, whereas VAEs only use an encoder and a decoder network, Generative Adversarial Networks (GANs) frequently have larger and more complex networks than Variational Autoencoders (VAEs). Due to the adversarial nature of GANs, training them may take longer than training VAEs since the generator and discriminator networks of GANs must learn to balance and reach a Nash equilibrium, which takes more time.

A potential benefit of VAEs is that they make sampling from the latent space simple. Simple sampling is used to create new images thanks to the encoder network in VAEs, which translates the input image to a latent space distribution. GANs, on the other hand, don't directly map input images to the latent space, which complicates sampling from the latent space.

Mode collapse, where the generator network generates fewer distinct modes of the data distribution than a variety of images, is a prevalent problem with GANs. The resulting photos may be monotonous or of poor quality. However, since their reconstruction loss does not expressly penalize sharpness, VAEs may experience hazy reconstructions.

In conclusion, while both GANs and VAEs are generative models capable of producing new data, they differ in terms of network size, training time, sampling from the latent space, and the caliber of the images produced. To decide whether approach is best for a given task, it's critical to carefully assess both approaches on a specific dataset.

4. Conclusion

In this study, we investigated the usage of two well-known generative models for creating fresh data samples: generative adversarial networks (GANs) and variational autoencoders (VAEs). On the MNIST dataset, we used both models to train them, and we assessed how well they produced high-quality photos.

After the GAN model had been trained, we used the trained generator network to produce fresh data samples and calculated the Inception Score to assess the quality of the resulting images. The GAN model was successful in capturing some of the MNIST dataset's salient features, although it had a comparatively low Inception Score when compared to other GAN models that had also been trained on MNIST.

We created new data samples after training the VAE model by selecting random samples from the learnt latent space, which were then decoded by the trained decoder network. To assess the quality of the generated images, we calculated the reconstruction error, which was relatively high when compared to cutting-edge VAE models trained on MNIST. Even though the VAE model was able to capture some of the essential elements of the MNIST dataset, the images it produced lacked the sharpness and clarity of those produced by the GAN.

In conclusion, new data samples generated using the MNIST dataset using the GAN and VAE models both demonstrated encouraging results. However, there is still opportunity for development in terms of producing images with greater sharpness and diversity. Future research might concentrate on investigating various model architectures and objective functions as well as training the models on larger and more complicated datasets to test their generalizability. Overall, this project offered insightful knowledge about the operation of two well-known generative models and their prospective use in producing fresh data samples for various purposes.

5. References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative Adversarial Nets". In: Advances in Neural Information Processing Systems 27. 2014, pp. 2672-2680.

[2] D. P. Kingma and M. Welling. "Auto-Encoding Variational Bayes". In: arXiv preprint arXiv:1312.6114. 2013.

[3] A. Brock, J. Donahue, and K. Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: arXiv preprint arXiv:1809.11096. 2018.

[4] T. Kim, M. Cha, H. Kim, J. Lee, and J. Kim. "Learning to Discover Cross-Domain Relations with Generative Adversarial Networks". In: arXiv preprint arXiv:1703.05192. 2017.

[5] A. Radford, L. Metz, and S. Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: arXiv preprint arXiv:1511.06434. 2015.

Software and Data

Softwares:

NumPy

Matplotlib

Scikit-learn

Python 3.6+

TensorFlow 2.0+

Data:

MNIST dataset (http://yann.lecun.com/exdb/mnist/)