Text To Image Generation In DCGAN and Stable Diffusion Model

Zhou Zhou Yunqing Zhu Norihito Naka
New York University Courant Institute of Mathematical Sciences
251 Mercer Street, New York, NY 10012-1185
zz3382@nyu.edu yz9661@nyu.edu nn1331@nyu.edu

Abstract

We explore the DCGAN model and stable diffusion model to implement text-to-image generation. We implement, train, and test both models' by using the CLIP ViT-B/32 model to implement the text embedding. We trained the DCGAN and the stable diffusion models on the MNIST dataset and used the string representation of digits 0-9 as text input. Moreover, we chose the inception score model inceptionv3 as a metric to evaluate the performance of models. Finally, we find the DCGAN model image quality performance is better than the stable diffusion models but the diffusion model can outperform the DCGAN model in time efficiency. In future work, we plan to extract the discriminator of the DCGAN model to the stale diffusion model to improve the denoising process performance.

1. Introduction

The text-to-image generation involves both natural language processing and computer vision. A successful text-to-image generation model converts writing language into images that correspond to the written context. This can be very effectively in helping people understanding textual concepts through images. Furthermore, artists and designers can also use this tool to present their ideas and creativity. Finally, text-to-image generation can be the foundation for implementing other AI drawing processes like Dalle2 and Imagen.

1.1. Related Work

Since 2015, when the diffusion model [1] was introduced, text-to-image generation has shown immense potential and growth. With the development of deep learning and associated hardware advancements, the training of deeper neural networks and more advanced architectures of neural networks have led to the performance of text-to-image generation to further improve. One important method for generating images is GAN (Generative Adversarial Nets) [2], which was introduced in 2014. The diffusion model has been gaining widespread usage and popularity since its introduction in 2015 due to its superior performance across vision tasks as well as stability when training [1] [3]. As two often used models in image generalization, both the GAN and diffusion models have their advantages and disadvantages. One of the most popular models now, Stable Diffusion is based on the diffusion model and has been showing great promise. Similar research has been conducted in 2022, researchers compared multiple methods for image generalization and different fundamental formulations including DDPMs, SGMs, and Score SDEs [4]. Moreover, in 2023, a method was introduced by Luca Guarnera, Oliver Giudice, and Sebastiano Battiato to discriminate between the image generated by GAN or by diffusion model [5]. As the two methods both show powerful abilities in generating the image, there are also some approaches trying to combine these two models. Researchers introduced a combination Diffusion-Gan in 2023 [6], which trains a diffusion model with GAN. This model uses the diffusion model to generate noise and utilizes the noise to train the GAN model, which can train the diffusion model without requiring a costly reverse diffusion chain. In addition, the advent of classified guidance and classifier-free guidance for diffusion models further motivates our exploration of ”mixture” models.

1.2. Proposed Approach

We explore the DCGAN model [7] and stable diffusion model to implement the text-to-image generation process. The DCGAN (Deep Convolutional Generative Adversarial Networks) can improve the stability and performance of normal GAN training. We implement the original DCGAN paper's generator and we use a binary classification as discriminator. In contrast, we also use the stable diffusion model [8] to implement the text-to-image generation process. We chose the U-net model [9] to generate images in the stable diffusion model and modify these models to be capable of our requirements. Then we use CLIP [10] ViT-B/32 model to implement the text embedding process for both models. In this process, these text embeddings will become the conditional signs to control related image output, so we add the cross-attention with intermediate layers into the stable diffusion model. In the MNIST dataset [11], we use numbers 0 to 9 as text input and generate related images. In the experiment process, the DCGAN model and stable diffusion model both generate reasonable results. The DCGAN model can generate clearer and less noisy images than the stable diffusion model. Based on the inception score of models [12], we find the DCGAN model performance is better than the stable diffusion model. We also plan to use the DCGAN discriminator model to add to the stable diffusion model in the denoising process to improve the model performance. Based on our experiment, we achieved our primary goal and explored the cutting-edge model in the text-to-image generation field.

2. Method

2.1. Stable diffusion model

The stable diffusion model is an improved version of the diffusion model, which originated from non-equilibrium thermodynamics and defines the image-generating process as forward and backward Markov processes. The stable diffusion model improves the speed of the diffusing model by using a compressed latent space in the forward process instead of the original image. Figure 1 indicates the diffusion model architecture. We use the conditional diffusion model to implement the text-to-images model. Kreis, Gao and Vahdat indicate $p_\theta(x_{0:T}|c) = p(x_T)\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, c)$. This is the probability distribution of generating a sequence of data points $x_{0:T}$. The $\prod_{t=1}^{T} p_\theta(x_{t-1}|x_t, c)$ is the product of conditional probabilities across all timesteps from T to 1, representing the reverse diffusion process. For the forward process, $p_\theta(x_{t-1}|x_t, c) = N(x_{t-1}; \mu_\theta(x_t, t, c), \sum_\theta(x_t, t, c))$ where μ_θ and $\sum_\theta$ are the mean and covariance predicted by the model for the next time step. The objective function is $L_\theta(x_0|c) = E_q[L_T(x_0) + \sum_{t>1} D_{KL}(q(x_{t-1}|x_t, x_0))||p_\theta(x_{t-1}|x_t, c) - \log p_\theta(x_0|x_1, c)]$. The $L_T(x_0)$ is a likelihood term for the data at the final timestep.

2.1.1 Time Embedding for Neural Network Integration

To ensure that the neural network captures temporal dependencies adequately, we use the time embedding technique. Instead of providing the network with a single linear time value, we express time as a combination of sinusoidal features. This enables the network to better respond to changes in time, facilitating the learning of a time-dependent score function $s(x, t)$. This process gradually adds noise to an image over a series of steps (forward process) and then learns to reverse this process to generate new images (reverse process). The time embedding tells the model how much noise was added at each step so it can learn to correctly reverse the process. Figure 2 indicates the forward and reverse process.

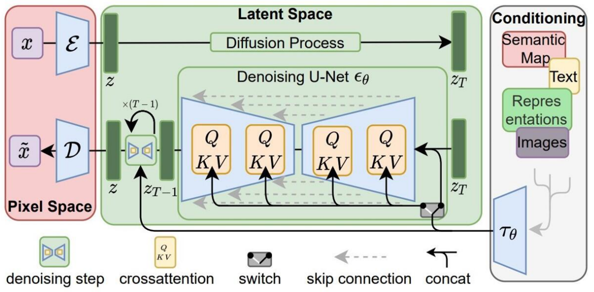

Figure 1. Latent Diffusion Models Overall Frame Diagram [13]

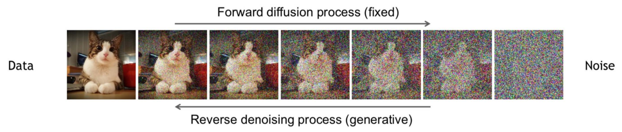
Forward diffusion process (fixed)
Data
Noise
Reverse denoising process (generative)

Figure 2. Denoising diffusion process includes forward diffusion process and reverse denoising process. [1, 8, 14]

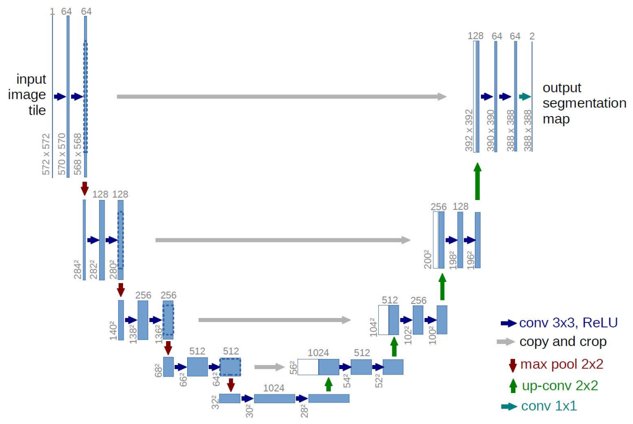

Figure 3. U-net model Architecture [15]

2.1.2 Defining the U-Net Architecture

We employ a U-Net architecture to approximate the score function and generate images. The U-Net structure is adapted to include time-dependent components and text-embedding, allowing the network to effectively capture features at different spatial scales. The Figure 3 indicates the basic U-net model architecture. We modify this model to accept the time embedding and text embedding.

Encoding and Decoding Layers The U-Net architecture consists of encoding and decoding layers. These layers involve convolutional and transposed convolutional operations, respectively, to handle feature extraction and generation. In this process, these layers of the U-Net allow it to be effectively aligned with the text and generated images.

Activation Functions and Normalization The architecture incorporates the swish activation function $\frac{x}{1+e^{-\beta x}}$ and group normalization $y = \frac{x-E[x]}{\sqrt{Var[x]+\epsilon}} * \gamma + \beta$ to enhance the expressive power of the neural network. The ϵ value is 0.00001.

Normalization and Output The final output is normalized by the standard deviation of the perturbation kernel, ensuring the generated images adhere to the desired distribution.

2.1.3 Alternative U-Net Architecture

Additionally, an alternative U-Net architecture is presented, highlighting variations in skip connection strategies. The choice between concatenation and addition of tensors from the down blocks allows for flexibility in the network's representation. The Figure 3 indicates the original U-net concatenation parts. Based on this process, we explore new skip connection strategies in the experiment.

Skip Connection Strategies Two strategies for skip connections are explored: concatenation and direct addition of tensors from the down blocks. The impact of these strategies on model performance is an avenue for further investigation.

Activation Function and Normalization Similar to the primary U-Net architecture, the alternative model employs the swish activation function and group normalization for effective feature learning.

2.1.4 Cross Attention Models

Word Embedding of Digits This section introduces the concept of word embedding for digits 0-9. In this model, we use an embedding layer to map digit indices to vectors. We add these text embedding into the stable diffusion model to generate related images.

Attention Layer Implementation In the stable diffusion model, we implement the cross-attention layer to connect text-embedding and generated images. The cross-attention layer includes the self-attention technique, which indicates the computation of query (Q), key (K), and value vectors (V), as well as the attention distribution and weighted average of value vectors. The self-attention can be described as $Attention(Q, K, V) = \frac{QK^T}{\sqrt{d_k}}V$. The text embedding will replace K and V in this progress. Cross-attention allows text embedding to interact with generated images and find out what they should pay more attention. Cross cross-attention layer is the most important part of the conditional model. In this case, we can use the semantic map, full text, representations, images and other conditionals to generate related images, like Figure 1.

2.1.5 U-Net Transformer

Transformer Block In the transformer block, we combine the cross-attention layer and feed-forward process. The block includes layer normalization and a two-layer MLP with a GELU nonlinearity. It can be represented as $GELU(x) = x * \phi(x)$, where $\phi(x)$ is the cumulative distribution function for gaussian distribution. In practice, we use $GELU(x) = 0.5*x*(1+Tanh(\sqrt{\frac{2}{\pi}}*(x+0.044715*x^3)))$. Moreover, the U-Net transformer represents a time-dependent score-based model built upon the U-Net architecture with added attention layers. The architecture includes encoding and decoding paths, skip connections, and Gaussian random feature embeddings for time.

2.2. DCGAN model

We use the deep convolutional GAN model by constructing and training the two major components: generator and discriminator. In addition to these two main models, we use CLIP ViT-B/32 model to construct the text embedding. In the DCGAN model, we use the text embedding to the generator and discriminator. The generator model can use text embedding to generate related images. The discriminator will try to distinguish whether the image is real or fake. We expected the generator could fool the discriminator, in this case, the generator can generate realistic images. The generator is $G(z, t; \theta_g)$ and the descriptor is $D(x, t; \theta_d)$ where t is the text embedding. In this case, the objective function of DCGAN model can be described as $min_G max_D V(D, G) = E_{p_{data}(x)}[logD(x|t)] + E_{p_z(z)}[log(1 - D(G(z|t)))]$. Furthermore, we also can describe the generator loss function $L_G = -\mathbb{E}_{z\sim p_z(z)}[\log D(G(z))]$ and discriminator loss function $L_D = -\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] - \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))]$.

2.2.1 Generator

We use the original DCGAN paper model to implement the generator similar to that represented in Figure 4 The generator architecture consists of a series of transposed convolutional layers, gradually increasing the spatial resolu-

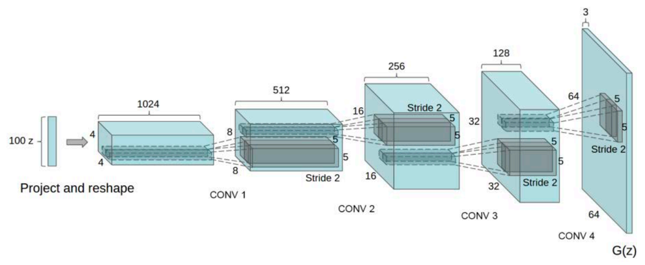

Figure 4. DCGAN Generator Model
[7]

tion. Each transposed convolutional layer is followed by batch normalization and ReLU activation, except for the output layer, which uses a hyperbolic tangent (Tanh) activation function. We modified this original model to be capable of our experiment requirement. In addition, we add the text embedding to the generator to control the final output images. We concatenate the generated images and text embedding, then we add this new image to the DCGAN generator model. We use ConvTranspose2d, BatchNorm2d, ReLU and Tanh to construct the generator model. Furthermore, we will pass the generated images to the discriminator. The generator will produce more realistic and related images after training.

2.2.2 Discriminator

The discriminator takes as input an image (x) and a text embedding (text). It is a binary classifier to distinguish whether real or fake (i.e. model generated). We use the text embedding in the discriminator as a condition. In this case, it can help the discriminator better distinguish between real and fake images. The discriminator not only distinguishes between generated or real images, but also determines whether the number generated in the image matches that in the textual context. The discriminator architecture consists of a series of convolutional layers, gradually decreasing the spatial resolution. Each convolutional layer is followed by batch normalization and LeakyReLU activation. The text embedding is processed separately through the Embedding model and then concatenated with the features extracted from the image. The combined features are processed through a final convolutional layer followed by a sigmoid activation function to obtain the discriminator's output, indicating the likelihood of the input being real or fake.

During training, the generator aims to produce realistic images to fool the discriminator, while the discriminator aims to correctly classify real and generated images. The generator is trained to maximize the likelihood of the discriminator being fooled, while the discriminator is trained to correctly classify real and generated samples. The GAN is trained using a combination of adversarial loss and possibly other auxiliary losses (e.g., text embedding consistency).

The models use hyperparameters such as `noise_dim`, `embed_dim`, and `embed_out_dim` to control the dimensionality of noise and embeddings. The choice of architecture and hyperparameters is likely motivated by experimentation and fine-tuning to achieve good performance on a specific dataset.

The performance of the GAN can be evaluated using metrics such as Inception Score, Frechet Inception Distance (FID), or visual inspection of generated samples. The GAN might be trained on a specific dataset, and the evaluation results could be presented to demonstrate the quality of the generated samples compared to real ones.

2.3. Training and Evaluation

The models are trained using a combination of adversarial loss and potentially auxiliary losses for text embedding consistency. The choice of hyperparameters, dataset, and training procedures are crucial factors influencing the model's performance. Evaluation metrics such as Inception Score, Frechet Inception Distance (FID), and visual inspection of generated samples were considered as methods for evaluating the performance across models. Inception Score was employed to assess the quality of the generated images.

3. Experiments

3.1. Dataset

We chose the MNIST dataset [11] as the training dataset rather than gathering data by ourselves. This dataset contains 60,000 training images and 10,000 testing images of handwritten digits from 0 to 9. In this dataset, each image pixel size is 28 $\times$ 28 with 1 channel. In the process of image generation, we utilize the labels as string inputs for the CLIP model.

3.2. Stable Diffusion Model

In the training process, we use A100 GPU to train 100 epochs in the stable diffusion model and each batch size is 1024. Furthermore, We choose the Adam optimizer with a learning rate of 0.0001. The stable diffusion model loss function can be described as $L = E[||score + std(t) + z||^2]$, where $std(t) = \sqrt{\frac{\sigma^{2t}-1}{2\log(\sigma)}}$. Figure 5 shows the stable diffusion model loss trend. In the forwarding process, we set the time step as 250.

3.3. DCGAN Model

3.3.1 Discriminator Model

We use adam optimization for both the generator and discriminator model. They both have a learning rate of 0.0002 and the beta coefficient for averages of gradient and square is 0.5 and 0.999. The discriminator model will accept images and text embedding as input parameters. The first output is similar to a normal GAN discriminator. This output represents the feature representation of the image as extracted by the discriminator's convolutional layers. The second output concatenates the primary output and text embedding, then the discriminator will use convolutional layers to indicate the probability that the given image-text pair is real. In the training process, we will use real images and related text embedding to get the real image features and text-related image features. Then we can get the BCE loss with real images and real labels. In contrast, the discriminator also will get the BCE loss with fake images and fake labels. Then we add the D_{real} loss and D_{fake} loss to get the final discriminator loss.

3.3.2 Generator Model

The generator model uses random noise and text embedding to generate images. In this process, we use the generated images and real labels to get the BCE loss. Then we use the L1 loss to compare the fake images with real images. We also use L2 loss to compare the discriminator's real image features and fake image features. We add the BCE loss, L1 and L2 loss to get the final generator model loss. We expected the generator would produce realistic images and the discriminator would not distinguish the real images and generated images.

3.4. Result and Discussion

In our experiment, we train the DCGAN model and stable diffusion model which both can generate reasonable hand-written images. However, there are also some notable differences between these two models.

One difference between the two models is the time involved in training. In our experiment, the learning speed of the stable diffusion model is much faster than that of the DCGAN model. Because of the generating process, the DCGAN model must spend time on both discrimination and generation, which is usually a costly process. Especially when the images are relatively simple, which keeps the backward process in the diffusion model easy to compute, the learning speed of the stable diffusion model can outperform the DCGAN model.

In the DCGAN model, because of the multiple models, the hyper-parameter tuning is more challenging. Our finding are consistence with those found in 2016, when researchers concluded that finding Nash equilibria in the GAN models is a very difficult problem [12]. In summary, the DCGAN model is both quantitatively and qualitatively more challenging due to the instability involved with finding an equilibrium convergence between the models involved. As a more straightforward model, the gradi-

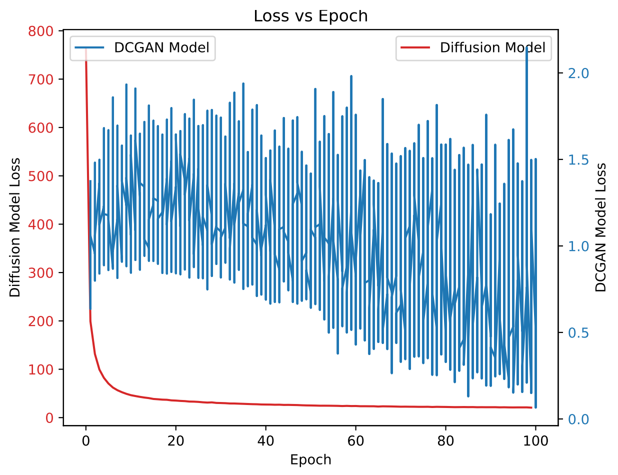

Figure 5. Comparison of Loss Convergence Over Epochs

ent descent process in a stable diffusion model is more stable. For diffusion models, fewer hyperparameters involved means less time spent tuning these in order to find an optimal setting that enables the model to perform well.

From figure 5, we see the loss decrease as the number of epochs increases. The Diffusion Model has a loss that is monotonically decreasing. The loss decreases the most during the first 10 epochs and then decreases slowly. On the other hand, the DCGAN Model more gradually decreases with a large variation in terms of concavity of the loss over epochs. While there is a general downward trend associated with the convergence of the model, overall, the loss curve of the GAN model is much more volatile. This may be caused by a number of factors including mode collapse, vanishing gradients, mismatched learning rates, and other hyperparameter initialization. Mode collapse is a common issue in training GANs, including DCGANs. Mode collapse occurs when the generator produces a limited diversity of samples, leading to a single mode in the distribution of the generated samples. This can cause the discriminator to become too confident in its predictions, leading to a decrease in the loss of the discriminator and an increase in the loss of the generator. This can result in a loss curve that is not monotonically decreasing. In terms of vanishing gradients, if the learning rate is too high, the gradients may vanish during backpropagation, causing the model to fail to learn effectively. If the learning rates for the generator and discriminator are not matched properly, the generator may be too powerful or too weak compared to the discriminator. These can both individually and collectively lead to a loss that does not decrease over epochs. The initialization of the weights in the network can significantly affect the training process. If the weights are not initialized properly, the network may not be able to learn effectively, leading to a loss that does not decrease over epochs. The batch size

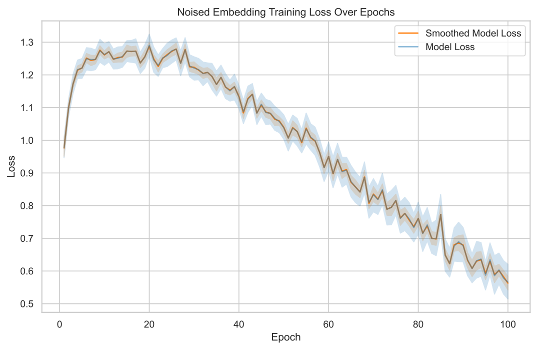

Figure 6. Model Loss Over Epochs Using Noised Image Discriminator

used for training can also affect the loss. If the batch size is too small, the estimates of the gradients may be noisy, leading to a loss that does not decrease over epochs. To address these issues, we tried various different weight initialization strategies, adjusted the learning rates, and used different batch sizes.

We add noise to the DCGAN discriminator model as the first step of determining whether it can be used in conjunction with the diffusion model for classified guidance. These results were measured for the images with Gaussian noise with standard deviation 0.1. We see from 7 that qualitatively the numbers are much more difficult to decipher. This takes inspiration from classifier guidance methods for diffusion models [16]. Our findings, however, were consistent with popular literature at the moment which leans towards classifier-free guidance as it applies to diffusion models [17]. When we used the noised image embedding with CLIP, we found that the model loss actually increased for the first approximately 10 epochs until it finally started to decrease as we can see through Figure 6. With the noise, the model seems less stable with the loss values not decreasing to the same levels as we saw as the baselines. If we were to apply this in the diffusion setting, we predict that the training process would be less stable and the model would perform worse, confirming some of the drawbacks of classified guidance.

Based on our experiment, we achieved our primary goal and explored the cutting-edge model in the text-to-image generation field.

Figure 8 shows the result of generating images by the diffusion model. For each number from 0 to 9, we generate ten examples using the MNIST-trained Diffusion Model. There were 250 steps involved with generating the image for each digit, which were then concatenated together to compose the final 10x10 image. Qualitatively, most generated digits are coherent and can be distinguished as well as differentiated without much prior knowledge. There are a number

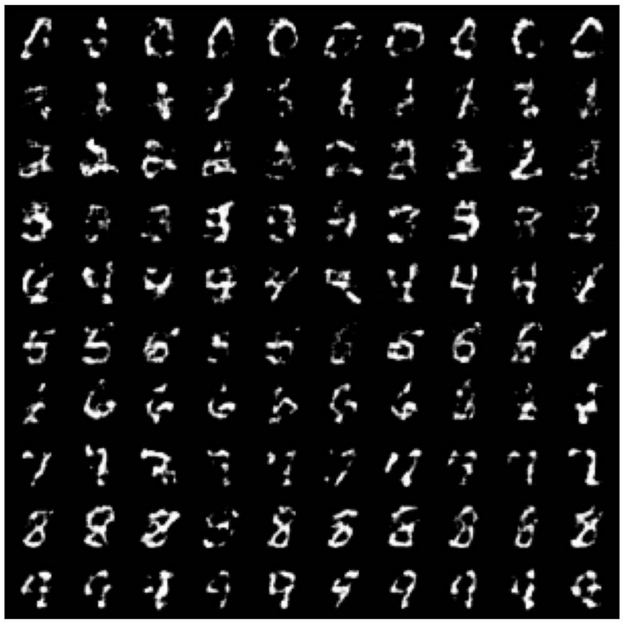

Figure 7. Model Output for Noisy Input Embedded Discriminator

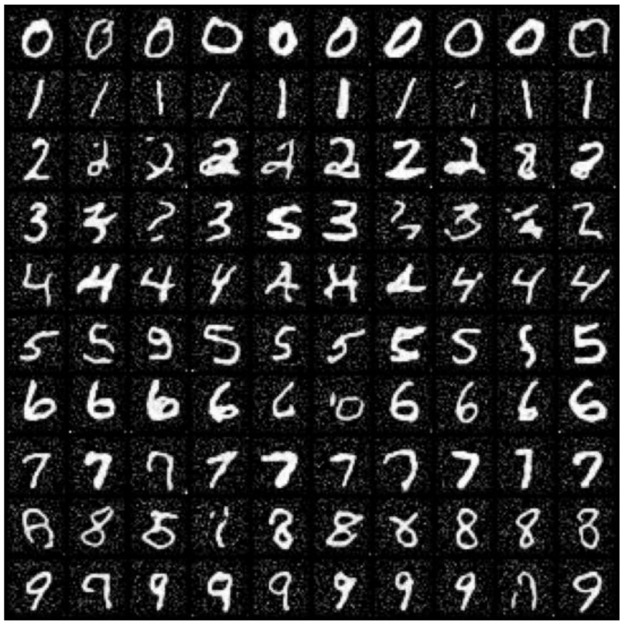

Figure 8. Examples of digits 0-9 generated based on the MNIST-trained Diffusion Model

of examples, however, in which similar digits appear under a row that does not correspond to the digit it is representing. For example, the ninth digit in the third row can also be interpreted as an eight. We see that as we increase the number of generation steps, the qualitative quality of the digit images improves. Moreover, the Inception Score eval-

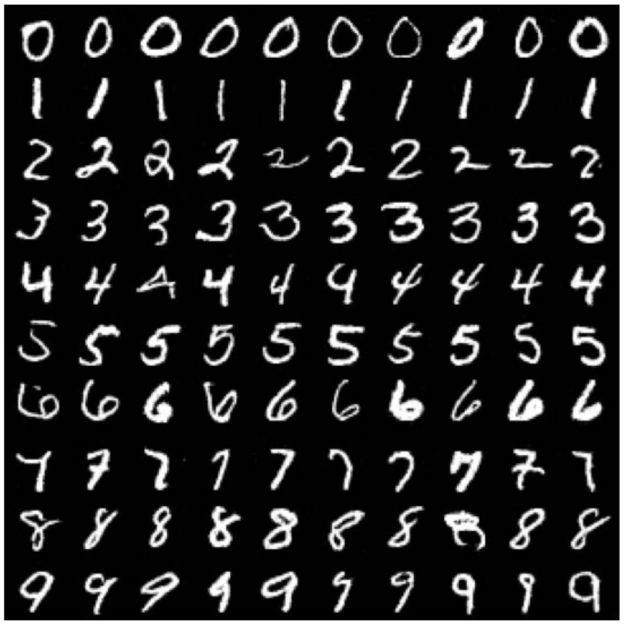

Figure 9. Examples of digits 0-9 generated based on the MNIST-trained DCGAN Model

Model	Inception Score	Standard Deviation
DCGAN	2.24	0.086
Diffusion	1.51	0.134

Table 1. Comparing the performance across models based on the Inception Score metric.

uation metric that we use only differentiates between real and fake images valid output images that may not match the expected output class would not be as highly penalized. Qualitatively, we see that some of the generated images are incomplete and have some noise in the background 8.

Figure 9 represents the DCGAN model final output from numbers 0 to 9. This image shows no noise in the background and many generated digits quality is better than the stable diffusion model. However, some numbers are still incomplete, for example, the last number 2.

The table 1 represents the inception score between the DCGAN model and diffusion model. We use the inception score model inceptionV3 as the metric to evaluate the model performance. In this case, the DCGAN model performance is better than the stable diffusion model.

4. Conclusion

In this project, we test and compare the two text-to-image generation models, the DCGAN model and the stable diffusion model. They both can be used to generate reasonable images but also have differences. Our experiment shows that the stable diffusion model is a more efficient way of image generation but poor image quality. The DCGAN model can generate a better quality number of images but this model is difficulty in adjusting parameters. We use the inception score to evaluate the performance of the model. In this progress, the DCGAN model can generate better images than the stable diffusion models. To improve the performance of models, we can combine the stable diffusion model and the DCGAN model advantages to implement the text-to-image model, for example, we plan to extract the discriminator of the DCGAN model to the stale diffusion model to improve the denoising process performance in the future.

References

[1] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. 1, 2

[2] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014. 1

[3] Ziyi Chang, George Alex Koulieris, and Hubert P. H. Shum. On the design fundamentals of diffusion models: A survey, 2023. 1

[4] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. Diffusion models: A comprehensive survey of methods and applications, 2023. 1

[5] Luca Guarnera, Oliver Giudice, and Sebastiano Battiato. Level up the deepfake detection: a method to effectively discriminate images generated by gan architectures and diffusion models, 2023. 1

[6] Zhendong Wang, Huangjie Zheng, Pengcheng He, Weizhu Chen, and Mingyuan Zhou. Diffusion-gan: Training gans with diffusion, 2023. 1

[7] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2016. 1, 4

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. 1, 2

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015. 1

[10] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. 1

[11] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012. 2, 4

[12] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016. 2, 5

[13] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models, 2022. 2

[14] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations, 2021. 2

[15] O. Ronneberger, P.Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. (available on arXiv:1505.04597 [cs.CV]). 2

[16] Vincent Tao Hu, Yunlu Chen, Mathilde Caron, Yuki M. Asano, Cees G. M. Snoek, and Bjorn Ommer. Guided diffusion from self-supervised diffusion features, 2023. 6

[17] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. 6