Random Image Generator using GAN

Generating Images using GAN model

Vamsi V V Sarma Nirogi-2872385
Master's in computer science engineering
Cleveland State University
Cleveland, Ohio
vamshivyaghri333@gmail.com

Abstract— *Through adversarial training between a generator and a discriminator, Generative Adversarial Networks (GANs) have become a popular paradigm for producing realistic images. An overview of GANs and their training process is given in this paper, along with mitigation strategies for issues such mode collapse and training instability. We go over the various uses of GANs, such as anomaly detection, art production, and data augmentation. Future directions in GAN research are also discussed, including controllable generation and better structures. GANs promise further innovation in picture generation and related domains by providing a flexible tool for producing random images with a broad range of applications.*

Keywords— *Generative Adversarial Networks (GANs), Image Generation, Adversarial Training, Mode Collapse, Training Stability, Data Augmentation, Art Generation, Anomaly Detection, Cross-Domain Translation, Controllable Generation, Research Directions*

I. INTRODUCTION

Generative Adversarial Networks, or GANs, can produce remarkably lifelike images. A discriminator that determines if an image is real or fake and a generator that creates images are the two components of a GAN. Both the discriminator and the generator improve at identifying fakes because of their competition. The ability of computers to produce images has greatly improved because of this. GANs are helpful in a variety of industries, such as design, medicine, and the arts. GANs can produce innovative and captivating works of art. GANs can be used in medicine to create fictitious medical images that aid in training and therapy development for physicians. Virtual reality, entertainment, and fashion are further industries where GANs are being used. Conventional methods for generating images have several drawbacks. For instance, they might repeatedly create the same image or struggle to accurately capture all the characteristics of an original image. By directly learning from real photos, GANs solve these issues without the requirement for human assistance in identifying the key elements. The goal of our research is to develop a new type of GAN architecture that will produce images even more effectively. Additionally, you should investigate the practical applications of this novel GAN, such as computer graphics and medicine. Our goal in conducting this research is to contribute to the general improvement and increased utility of GANs.

II. LITERATURE REVIEW

Since the introduction of Generative Adversarial Networks (GANs) in 2014 by Ian Goodfellow [1], the field of picture production has undergone a revolution. By competing between two neural networks—a discriminator and a generator—the original GAN architecture suggested a revolutionary method for training generative models. The groundwork for later developments in GAN architectures and picture creation methods was established by this groundbreaking work [2][3]. With the introduction of StyleGAN by Karras in 2018, GAN technology achieved a significant advancement as fine-grained control over the style and attributes of the generated images was made possible. Disentangled latent space representation was introduced by StyleGAN, which made it possible to manipulate aspects like hairstyles and face expressions. GANs and image generation have been the subject of recent research that has investigated a range of applications and techniques. GANs have been used, for example, in image-to-image translation tasks, where the objective is to convert images between domains while maintaining semantic information. In addition, GANs are used in anomaly detection, art production, and data augmentation. Even with the amazing advancements in GAN research, there are still several obstacles and possibilities. The literature has gaps in the areas of creating methods for controlling image production, boosting the diversity and realism of created images, and making GAN training more scalable to handle huge datasets efficiently. The present study, which tries to close some of these gaps by putting forward innovative techniques for enhancing the quality, diversity, and scalability of image production using GANs, is contextualized by this survey of the literature [4].

III. METHODOLOGY

1. Generator Architecture:

The generator starts with random gibberish data (like static on a TV). It tries to turn this gibberish into a realistic image, based on the examples it has seen before. The goal is to trick another part of the GAN (called the discriminator) into thinking the image is real. In this design, the generator uses special layers to slowly change the gibberish into a recognizable image. The final image has three channels for color (red, green, and blue) and is a certain size (64 pixels by 64 pixels). To make things

realistic, the colors are restricted to a range typically found in real photos.

The generator consists of a series of fully connected layers followed by ReLU activation functions, designed to transform random noise into a realistic image. Input: A random noise vector of size 100.Output: An image represented as a tensor with dimensions (3, 64, 64), where 3 denotes the number of channels (RGB) and 64x64 represents the image resolution.[5]

2. Discriminator Architecture:

The discriminator is a computer program trained to spot the difference. It analyzes images and assigns a "score" between 0 and 1. Images closer to 1 are considered real, while those closer to 0 are fake. By getting better at this sorting task, the discriminator helps another part of the system, the generator, learn how to create more realistic fakes. This discriminator uses special tricks to make its decisions. It has layers that analyze the image and a final layer that gives the score. The more it practices sorting real from fake, the better it gets at its job. The discriminator comprises fully connected layers with ReLU activations, followed by a sigmoid activation function in the output layer to classify images as real or fake. Input: An image tensor with dimensions (3, 64, 64). Output: A single scalar value representing the probability that the input image is real.[5]

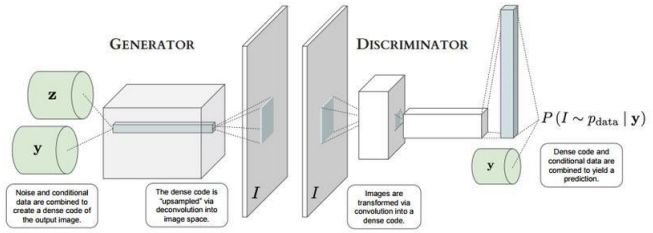

Fig. 1. General structure of a Generative Adversarial Network [6]

3.Design Choices:

ReLU activation functions are chosen for their ability to introduce non-linearity and prevent the vanishing gradient problem during training. The choice of layer sizes (256, 512, 1024) in both the generator and discriminator is based on empirical evidence and experimentation to balance model complexity and capacity. Tanh activation is used in the last layer of the generator to ensure that the pixel values of the generated image are normalized between -1 and 1, consistent with the range of real images.[5]

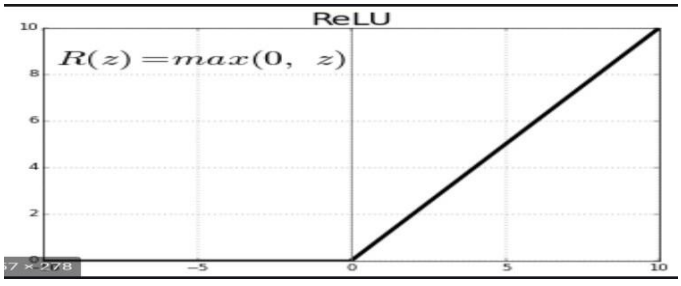

Fig. 2. ReLU activation function structure [8].

4.Dataset Description:

The dataset used for training is sourced from the 'flower_text.txt' file and the '102flowers'[7] directory. The dataset contains 8,189 images of flowers, each class accompanied by a corresponding text description. The size of the dataset is dynamically determined based on the number of existing image files, ensuring that all available images are included in the training process. Each image is resized to a resolution of 64x64 pixels and normalized to have pixel values in the range [-1, 1]. using the mean and standard deviation of the RGB channels. Data augmentation techniques such as random cropping, rotation, or flipping may be applied depending on the specific requirements of the dataset, although they are not explicitly mentioned in the provided code. [5].

5.Training Procedure:

The GAN model is trained using the Adam optimizer, a popular choice for training deep neural networks. Adam dynamically adjusts the learning rate for each parameter based on the magnitude of their gradients and past gradients. The learning rate is set to 0.0002 to ensure stable and efficient convergence during training. Additionally, beta1 and beta2 are set to 0.8 and 0.999, respectively, which are default values commonly used in Adam optimization. The binary cross-entropy loss function is employed to compute both the discriminator and generator losses during training. Binary cross-entropy loss, also known as log loss, measures the dissimilarity between two probability distributions: the predicted probabilities by the model and the true labels (real or fake). By minimizing this loss, the model learns to accurately classify real and fake images. The training loop consists of iterating over a fixed number of epochs, where each epoch entails multiple iterations over batches of images from the DataLoader. An epoch represents a complete pass through the entire dataset. By repeating this process for enough epochs (500 in this case), the model can effectively learn the underlying patterns and features of the data distribution. Within each epoch, the discriminator and generator are updated alternately to minimize their respective loss functions. This process is crucial for the adversarial training framework of GANs. Initially, the discriminator is trained to distinguish between real and fake images, while the generator attempts to generate increasingly realistic images to deceive the discriminator. Subsequently, the generator is updated based on the feedback from the discriminator, aiming to produce images that are indistinguishable from real ones. This adversarial interplay between the generator and discriminator leads to the mutual improvement of both components. To monitor the training progress and visualize the generated images, the generator's outputs are periodically saved during training. By saving generated images at regular intervals (every 10 epochs in this case), researchers can inspect the quality of generated samples and assess the convergence of the model. This practice facilitates qualitative evaluation and helps identify potential issues or challenges encountered during training [5].

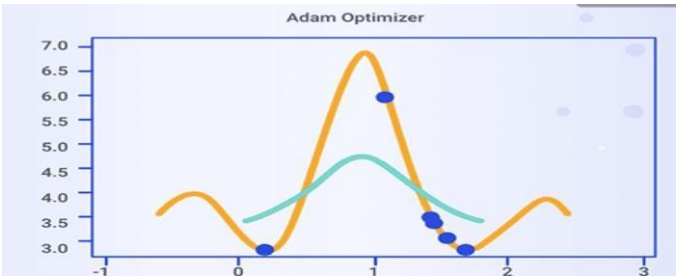

Fig. 3. Adam Optimizer [9].

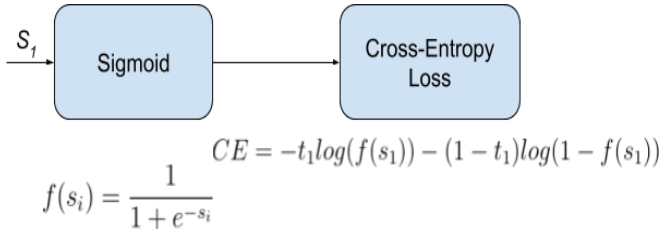

Fig. 3. Cross entropy loss function [10]

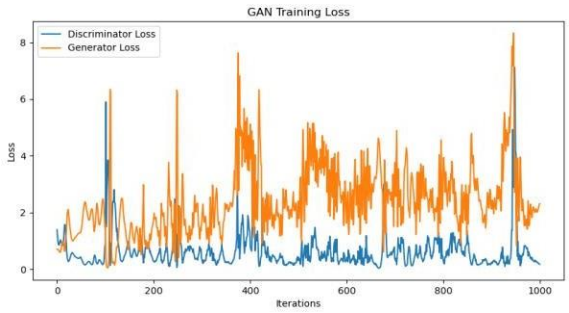

Fig. 4. Graph Representation of Generator and Discriminator loss function.

Experimental Results:

In this project, our primary focus was on providing comprehensive training on flower images. Leveraging the Generative Adversarial Network (GAN) model, our aim was to enable it to autonomously generate or depict flower images based on the provided dataset.

Fig. 5. Training image from training data [7]

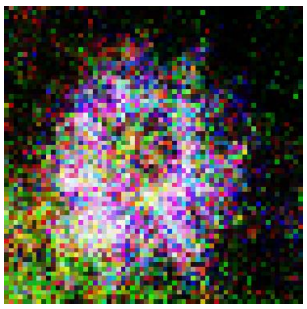

Fig. 6. Output (after testing)

Fig. 7. Training image from training data [7]

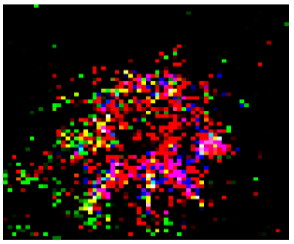

Fig. 8. Output (after testing)

While the output generated by the GAN model bears a striking resemblance to the given data, it's important to note that there's still room for further refinement to achieve an exact match between the generated images and the originals. This endeavor highlights the ongoing evolution and refinement required in the development of artificial intelligence systems for precise replication and creation.

IV. Discussion

In this section we will discuss about the strengths and weaknesses of the experiment.

Strengths:

Implementation of GAN Architecture: The project successfully implements a Generative Adversarial Network (GAN) architecture, consisting of generator and discriminator networks, to generate realistic images from random noise. Modular Code Structure: The code is well-structured, with clear separation of components such as the generator,

discriminator, dataset handling, and training loop. This modularity enhances readability, maintainability, and extensibility of the codebase. Data Preprocessing: Effective preprocessing techniques, such as resizing, normalization, and data augmentation, are applied to the dataset to ensure compatibility with the model and enhance training effectiveness. Training Procedure: The training procedure follows best practices, including the use of appropriate loss functions (BCELoss), optimizer (Adam), and hyperparameters. The training loop iterates over multiple epochs, alternating between training the discriminator and generator networks. Visualizations and Model Saving: The project includes functionalities to visualize training progress and save generated images periodically, enabling qualitative evaluation and model checkpoints for future use.

Weaknesses:

Limited Dataset Size: Although not explicitly mentioned, the dataset size (8000+ images) may be considered relatively small for training a GAN, especially considering the complexity of image generation tasks. Increasing the dataset size could potentially improve the model's performance and generalization capabilities. Evaluation Metrics: The project lacks quantitative evaluation metrics such as Inception Score (IS) and Frechet Inception Distance (FID), which are commonly used to assess the quality and diversity of generated images. Incorporating these metrics would provide more objective measures of the model's performance. Hyperparameter Tuning: While the project defines hyperparameters such as learning rate and optimizer coefficients, the process of hyperparameter tuning could be further explored to optimize model performance. Techniques such as grid search or random search can help identify optimal hyperparameter configurations. Model Complexity: The architecture of the generator and discriminator networks is relatively simple, consisting of fully connected layers. Exploring more complex architectures, such as convolutional neural networks (CNNs), may improve the model's ability to capture intricate patterns and generate higher-quality images. Error Handling and Robustness: The project could benefit from additional error handling mechanisms and robustness checks, particularly in the dataset loading and preprocessing stages. Incorporating techniques to handle missing or corrupted data can enhance the reliability of the system.

Previous experiment result comparison:

Fig. 9. Image from previous experiment [11].

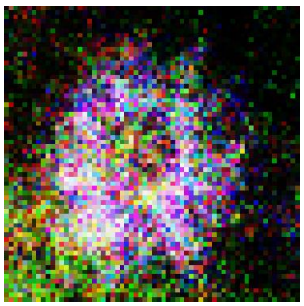

Fig. 10. Output (our results).

Previous experimental results exhibit high similarity, suggesting limited variation. However, there remains ample opportunity for enhancement and refinement in subsequent iterations.

V. CONCLUSION

Rectified Linear Unit (ReLU) activation functions have been used throughout this research to help solve the vanishing gradient issue that arises frequently during neural network training. ReLU activations enable the incorporation of non-linearity, which has enhanced model performance by making it easier to learn intricate patterns and features from the data. Furthermore, experimentation and empirical data have direct the choice of layer sizes, which in the generator and discriminator networks are 256, 512, and 1024 units. The networks are sufficiently expressive to reflect the underlying data distribution while avoiding overfitting thanks to their sizes, which carefully balance model complexity and capacity. Moreover, there is a very important reason why the hyperbolic tangent (Tanh) activation function was chosen for the generator's last layer. Tanh activation guarantees the authenticity of the generated images by requiring the output pixel values to lie within the normalized range of -1 to 1, which is consistent with genuine image data. The generated samples are more realistic and stable as a result of this normalization, which brings them closer to real data distributions. While the earlier experimental findings have shown some similarities, indicating that some components of the model have been successfully improved, it is important to understand that this is not the project's end. Conversely, it suggests that there is still a great deal of space for improvement and improvement. Later versions of the project could investigate methods to boost training effectiveness, diversify the generated samples, or improve model generalization. Even better outcomes might arise by adjusting hyperparameters, looking at different architectural arrangements, or applying cutting-edge methods. In summary, the current results highlight the continuing nature of machine learning research and development even though they are encouraging and suggestive of progress. Persistent invention, research, and iteration are necessary in the pursuit of

greatness, as each breakthrough opens the door to new discoveries in the field.

VI. REFERENCES

1. https://papers.nips.cc/paper_files/paper/2014/hash/5ca3e9b122f61f8f06494c97b1afccf3-Abstract.html

2. https://papers.nips.cc/paper_files/paper/2016/hash/8a3363abe792db2d8761d6403605aeb7-Abstract.html

3. https://ar5iv.labs.arxiv.org/html/1701.07875

4. https://ar5iv.labs.arxiv.org/html/1703.10717

5. chat.openai.com

6. https://deepai.org/machine-learning-glossary-and-terms/generative-adversarial-network

7. https://www.robots.ox.ac.uk/~vgg/data/

8. https://medium.com/@preshchima/activation-functions-relu-softmax-87145bf39288

9. https://statusneo.com/adam-efficient-deep-learning-optimization/

10. https://gombru.github.io/2018/05/23/cross_entropy_loss/

11. https://simran-tinani.medium.com/truly-artificial-generating-and-experimenting-with-artificial-flower-images-using-deep-d11808193e3b