Adversarial Insights: Leveraging GANs for Handwritten Digits

Ashutosh V. Patil

Student, SCTR's Pune Institute of Computer Technology, IT Engineering Department Pune, Maharashtra, India
ashutoshpatil359@gmail.com

Abstract

In recent years, generative adversarial networks (GANs) have become an AI method, especially for generating the real synthetic data. This paper studies the performance of GANs in generating the mathematical algorithms using MNIST dataset as a benchmark for evaluate the performance standards. So we present a novel method using GAN architecture, which combines processes to improve quality and variety. Our approach incorporates rigorous training methods and optimization strategies to address issues such as model crashes and instability during training. We conduct a comprehensive analysis of the results obtained from qualitative and quantitative analysis, including indicators such as scores. Additionally, we discuss the implications of our results for the use of data augmentation and its potential to improve digital experiences. This work demonstrates the potential of GANs in synthetic data generation and paves the way for future research in developing the models for practical applications.

Keywords : GAN, MNIST, loss, model, generator, discriminator

1. Introduction

Generative adversarial networks (GANs) have become a widespread in areas of ML , especially modelling. Published bulan Good-fellow and his friends of college[1] in previous year , GANs feature 2 neural networks (Creator(G) and Evaluator(D)) that are rooted in game theory. The function of the generator is to create a synthetic data model similar to the target classification, while the controller determines whether the model is derived from a real data or the synthetic data from the generator. This attack process causes both networks to continue to improve and allowing the generators to create more accurate models over time. [1] Video generation, Text generation and many ore. By learning how to classify the information of the training set, GANs can generate new events that exhibits similar properties, making them useful for tasks that requires data synthetic quality. [1]

Importance of handwritten digits

Reading various code is a classic problem in computer vision and pattern recognition and has important applications. Processes that can recognize postal codes are important[2] in areas such as:

- Postal automation
- Data Entry automation
- Banking
- Historical Document analysis

The MNIST dataset has collected examples of basic number's for research in this area. However despite the availability of this information performance problem persists especially when dealing with changes in text, sound etc. that would affect real-world[1] recognition.

The impact of GANs in data augmentation and artificial data generation

In applications of neural networks in handwritten digits useful for enhancing a learning process of ML models. This architecture can reduce the limitations of small datasets by generating additional training models, thereby improving the models ability. GANs can be used specifically for the following purposes :

- Data Augmentation
- Synthetic data generation

By exploring these points, this paper contributes to a border understanding of the suitability of GANs for generating synthetic data, particularly for code. [2] We also aim to provide reference points in upcoming studies and encourage through further exploration in GANs as a tool to improve machine learning in various domains.

Literature Survey

In [1] The paper entitled "An Overview of GANs" here Ian Goodfellow specifies the Gan machine learning framework in 2014. It consists of an adversarial trained discriminator and generator. Even though GANs have demonstrated a lot of promise in producing lifelike pictures, videos, and 3D objects, their training can be erratic and vulnerable to problems such as mode collapse. Numerous extensions have been created to enhance output quality and stability, including Conditional GAN (CGAN) and Wasserstein GAN (WGAN). Additionally, GANs are used in face recognition, 3D object reconstruction, and privacy protection. Despite their achievements, there are still issues with handling discrete data and increasing training efficiency.

In [2] the paper entitled "Enhancing The productivity of Image Classification Using Deep Learning techniques" Image classification has greatly improved thanks to deep learning, especially when CNNs are used. Pre-processing methods are essential for increasing model accuracy. Random cropping, grayscale conversion, and normalization are some methods that improve data quality and lessen overfitting. Research indicates that techniques such as CLAHE increase medical image classification accuracy by 5%. Excessive pre-processing, however, can obscure crucial information. Pre-processing methods must therefore be carefully chosen and optimized for various datasets and architectures.

In [3] the paper entitled "Generative Adversarial Networks: A Brief History and Overview" A generator and a discriminator that operate antagonistically to enhance data generation make up the deep learning model known as Generative Adversarial Networks (GANs), which was first presented in 2014. With the advent of DCGANs and StyleGAN, which improve image quality and generation control, GANs have seen substantial advancements. They are frequently employed in tasks such as conditional image creation and image generation. Research is still being done to increase training stability and efficiency because GANs have issues with mode collapse and training instability.

In [4] the paper entitled "GAN's : An summary of theory and its applications" Since their introduction by Ian Goodfellow in year 2014,(GANs) have transformed a number of industries, most notably machine learning and image generation. They can be made up of two neural models —a discriminator and a generator—that cooperate in enhancing the calibre of the data that is produced. GANs have been effectively used in fields like face detection, medical image synthesis, and 3D object generation. Notwithstanding their achievements, problems like mode collapse and training instability still exist, but research is still being done to improve and broaden their uses in sectors like texture transfer, traffic control, and healthcare.

In [5] the paper entitled "Exploring deep ConvNets" in biometric systems: a survey study" Convolutional neural networks (CNNs) and GANs are combined in deep ConvNets (DCGANs) to create high defination synthetic pics. DCGANs have found extensive use in biometric machines, particularly in the creation of artificial biometrics such as facial images and fingerprints. This method has demonstrated promise in data augmentation, spoof detection, and enhancing the precision of biometric system recognition. According to recent research, DCGAN can produce realistic biometric samples, enhancing cybersecurity by producing reliable training datasets. But issues like adversarial robustness and image quality continue to exist, which motivates more study into biometric applications.

2. Background and Related Work

Adversarial Generative models (GANs) are revolutionizing the field about generative modeling by introducing techniques for generating the knowledge from adversaries. The GAN architecture has two vital models: The Producer (G) and The Evaluator (D).

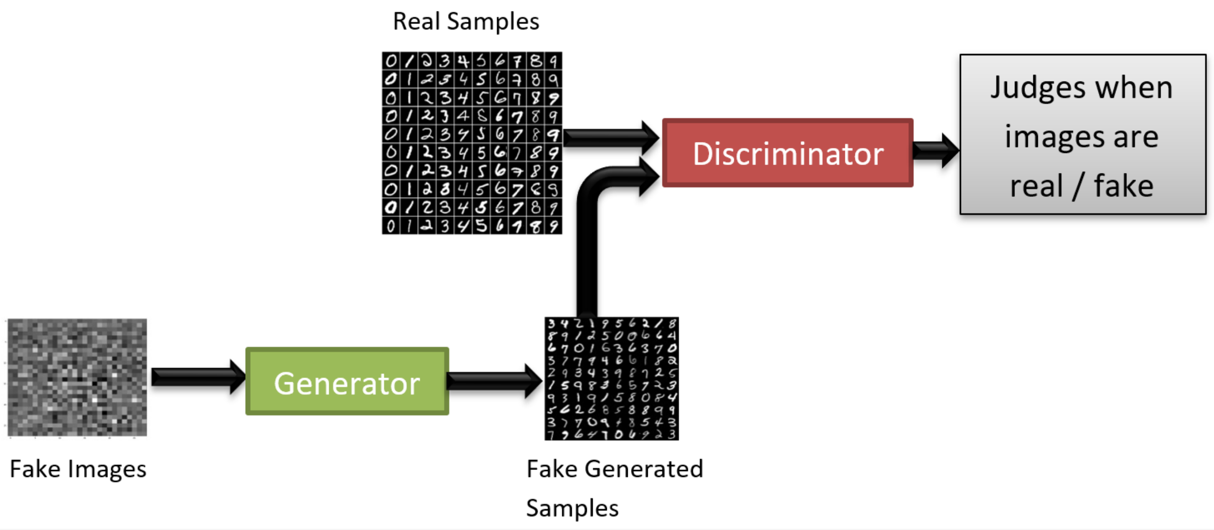

Fig. 1. Generative Adversarial Networks

Generator (G) :

The generator is tasked with producing simulated data models. Here firstly a random noise is produced to an image and then it converts into a data label model that mimics the distribution of the actual data. Generator architecture typically includes multilayer neural networks and techniques such as switching plugins and generating high throughput are often used. [4]

Discriminator (D) :

The evaluator (D) in contrast functions as binary classifier that distinguishes between genuine and generated data samples. It receives real data from the training set and fake data created by the producer (G) This model typically resembles ConvNets employing various convolutional methods to extract features from the input data. [4]

Training process & performance loss [3] of a network is expressed as min-max game between producer and evaluator and is mathematically expressed as:

$$\min(X) \max(A)\, V(A, X) = A_{x\sim Pdata}0\,(x)\,[\log A(x)] + A_{Z\sim P_Z}0\ \ (z)[\log(1 - A(X(z)) \ldots(1)$$

Here, $Xdat(x)$ is real data $Pz(z)$ is a noise distribution. The goal of the generator is to minimize the probability that D is identified as a false model, while the goal of A is to increase the true accuracy of different false models. This is an important process. Initially, the original GAN formulation used binary cross-entropy loss. [3] However, overtime it has been improved to improve he stability and integration.

For example, Wasserstein GAN(WGAN) provides better gradients for training by replacing the unemployment model with the earths mover distance.

GAN was proposed by GoodFellow et al. It represents a significant milestone in the filed of generative models. [3] The original GAN architecture has proven effective in generating synthetic images and forms the basics for many applications such as:

- Image rendering: GANs are widely used to generate real-time renderings in many domains, including natural scenes, animals and human faces. Their ability to create good models makes them a popular choice for computer programming. [8]
- Text –to-image Synthesis: It involves using GANs to create images based on textual descriptions, allowing for the linking of similar images across attributes.
- Image Upscaling: GANs are used to enhance the resolution with low-quality images and achieve excellent results in creating best-quality images.

The following graph compares the accuracy of different GAN (Generative Adversarial Network) frameworks implemented in various papers :

a) Vanilla GAN: The standard GAN model.
b) Conditional GAN: A GAN that incorporates additional information to guide the generation process.
c) DCGAN: A ConvNets that uses conv layers for both the producer and evaluator.
d) Wasserstein G**AN:** A GAN that uses the Wasserstein distance as a reduce function, leads to a good training model.

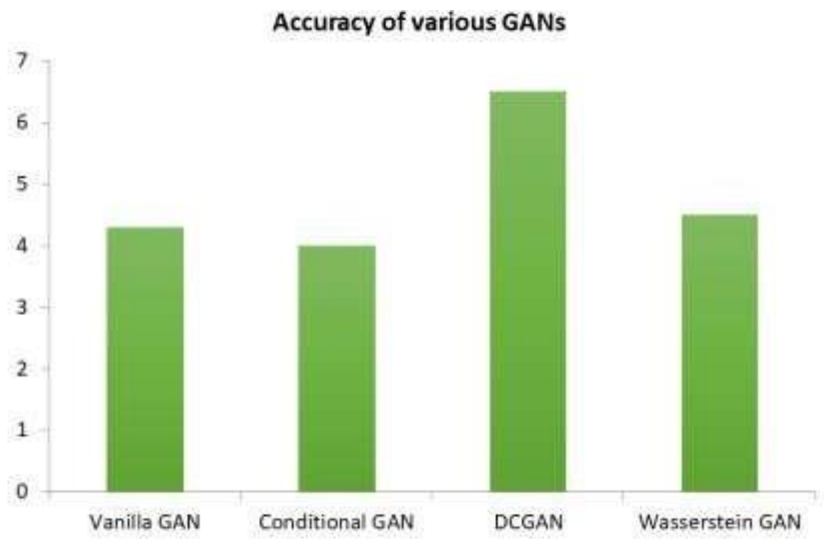

Fig. 2. Comparing Various GANs

3. Notable Studies Focusing on Handwritten Digit Generation

The applications of GANs to create a code generators has attracted considerable attention especially in the context of MNIST dataset, which serves as a benchmark for code generators. Some notable works include:

- DCGAN: Radford et.al (2015) introduced the DCGAN architecture that uses convolution techniques in both generator and discriminator. This architecture is most effective in generating high-quality code using deep and carefully selected architectures.
- CGAN: Mirza and Osindero (2014) proposed a conditional GAN that allows the creation of models conditioned on specific domains. This method is particularly effective in creating digital models corresponding to specific codes, thus increasing control of the output.

- CycleGAN: Zhu et.al (2017) taught CycleGAN to transform images from one domain to another without sampling. Although mainly used for image-to-image transformation, the principles have been adapted to generate code from other types of representations. [8]

Variants of GANs Used in Similar Tasks

The versatility of GANs has led to the development of many variants for specific applications. Some of the most promising developments in the development of code algorithms are:

- Deep Convolutional GAN (DGAN) : This evolution uses a deep network that can generate high-resolution images. The architecture consists mainly of convolutional process, batch normalization and ReLU activations allowing it to capture complex details in design.
- Wasserstein GAN (WGAN) : WGAN solves the security training problem encountered in traditional GN by using Wasserstein distance as a function loss. This approach has shown better integration and has been successfully used to generate high-quality graphs including code.
- Development of GANs: Karras et.al (2017) proposed progressive GANs begin with low- resolution images and progressively enhance them to higher resolution by the period of training. This method gives good results in generating good examples and can be adapted to coding.
- StyleGAN: A new architecture that separates patterns and content in graphic design. This approach enhances the control over more produced images, making it possible to compete to generate different numbers.

Thus, the development of GANs has led to significant advances in coding and a variety of architectures and techniques have been explored. [8]This background sets the stage for the reminder of this article, where we will examine the use and evaluation of GAN to create a handwritten digits an to identify it.

This section explains the dataset used, the structure of GAN model (producer and evaluator) , and the
[8] details in the training process including hyperparameters and optimization techniques.

4. MNIST dataset Overview

The MNIST dataset is a standard set of data specially in the field for ML and DL with any AI frameworks, specifically designed for manual written tasks which consists of hand-written samples. It consists of 70,000 grayscale pics of numbers which are handwritten, each of compatibility of (28,28) px. The dataset is divided into (10,000) test photos with digits ranging from 0-9. [11]

Fig. 3. MNIST Handwritten Dataset

MNIST dataset is ideal or generative tasks, such as those performed by GANs due to its simplicity relatively small size and widespread use, [11]allowing for easy comparison with other generative models. It has been used extensively for testing and validating models like classification and image generation.

Standardization: The values in MNIST dataset pics ranges in 0 to 255. To make the training more[11] sufficient and to stabilize GAN training the px values to be normalized in range of [-1,1] . This is done using the formula:

$$Normalized\ pixel\ value = \frac{orignal\ pixel\ value - 127.5}{127.5} \ldots (2)$$

So, this normalization helps GANs especially with activation functions like tanh which works well with inputs in the range of [-1,1].

- Reshaping: The images are reshaped to 28x28x1 where 1 represents the single channel for grayscale. This ensures compatibility with convolutional layers in the GAN architecture, which expect a 3D input for each image (height, width, channels).
- Batching: The dataset is split into batches of size BATCH_SIZE (e.g., 128 or 64) for efficient processing. Batching allows the model to update its weights based on a subset of the data at each step, fasting the model fitting and ensuring good generalization.[11]

Importance of MNIST

The MNIST dataset was chosen for this study due to following points:

- Standard Benchmark: Most widely used dataset for image evaluating image and classification methods.
- Simple & Preprocessing : The data in a dataset is well structured & which requires minimal preprocessing compared to real world dataset.
- Training Efficiency : The dataset is relatively small allowing train ing and optimization of generative models.

Comparison for other datasets

Table 1. Comparison for Possible datasets

Dataset	No. of Images	Image Size	Classes	Complexity	Suitable GAN's
MNIST	70,000	28 x 28 px	10	Low	YES
EMNIST	814,255	28 x 28 px	62	High	YES
NIST SD19	814,255	Varies	62	Very High	NO
uzushiji - MNIST	70,000	28 x 28 px	10	Medium	YES

Since the primary objective of this research is checking the GAN's for H.D so that the MNIST is well suited for the specified structure.

5. GAN Architecture :

The Producer (G) and the Evaluator (D) are the parts of GAN model. Each component has a different architecture for generating and classifying handwritten digits.

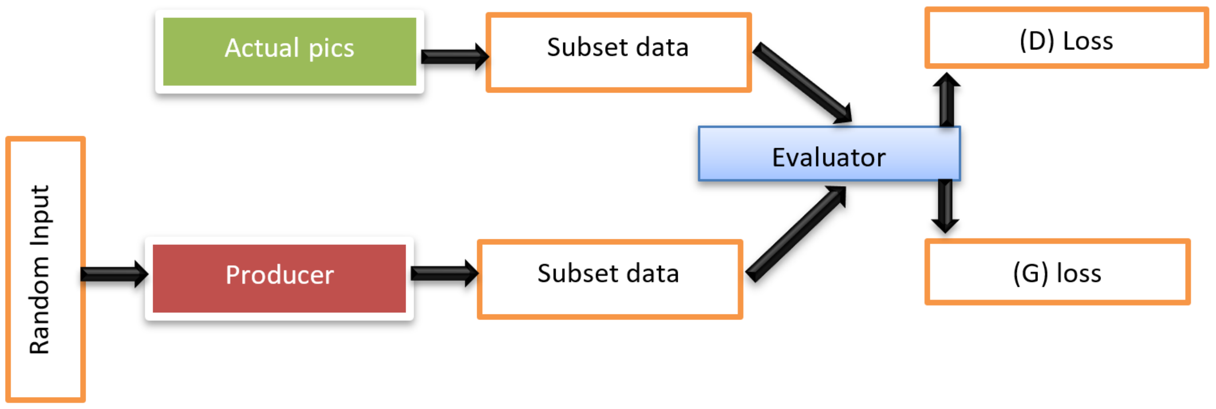

Fig. 4. Working Architecture

5.1. Generator Model

The generator is responsible for creating realistic images of handwritten digits from random noise. A normal distribution of noise is mapped to a synthetic image. The generator uses a series of deconvolutional layers to upsample the noise in to a full-size image.[7]

The Architecture layers:

- The input is a 100 dimensional noise vectors. There is a random sample from a Z-distribution. A fully connected terms translates the 100 dimensional data into 7 x 7 x 128 feature map. The base for generating the image is here.
- The fully connected layer output is fed into a sequence of de-convolutional layers in Upsampling the feature pic to 28 x 28. Each [7]convolutional strata is followed with rectified linear unit activation

function and a batch serialization.

- strata 1: Upsampling to 14 x 14 x 128
- strata 2: Upsampling to 28 x 28 x 64
- strata 3: producing the final output image of shape 28 x 28 x 1 which uses tanh activation function to ensure the px values in proper ranges.

Key architectural features

- Normalization is applied after each deconvolutional layer.
- The output strata is where tanh comes in strata .
- The tanh function makes sure the output matches the normalized values.

5.2 Discriminator Model

The discriminator separates generated (fake) images from real ones by acting as a binary classifier. It receives a 28x28x1 image [7,11,5]as entry and returns a prediction indicating whether the pic is authentic (produced by the generator) or fraudulent (derived from the training dataset).

The Architectural Layers:

- Layer of Input: A 28x28x1 real MNIST image or a phony image produced by the producer.
- Convolutional Layers: The Evaluator is made up of multiple ConvNets that extract attributes from the entry image with gradually downsampling it.
- Absolutely staked layer: The result of the final ConvNets is exceeded through a totally related strata, which produces a transformer output predicting that the input picture is actual or fake. A sigmoid activation feature is carried out to output a chance rating between 0 and 1. [7][11][2]

Key architectural features:

- Leaky ReLU: Used rather than normal ReLU in the discriminator to keep away from lifeless neurons and permit small gradients for negative inputs.[7][11]
- No Batch Normalization: the discriminator usually does now not use batch normalization, as it is able to clean out the learning signal vital for distinguishing between actual and fake pix.

5.3. Training the optimization Strategies

- Learning Rate Scheduling
- Batch Normalization & Spectral Normalization
- Adam Optimization with tuned B1 and B2 parameter

- Key Contributions:
- More Stable Training : Gradient penalty & Spectral Normalization and prevent stability
- Higher Quality Digits : Improved Discriminator training production for sharper images
- Better mode Collapse Prevention : Noise Injection and loss function where tuning ensures diversity in generated images
- Lower FID & Higher IS Scores **:** Outperforms existing GAN's on the MNIST Dataset.

6. The Optimization Process

The instruction manner in GAN is antagonistic which means that the producer and evaluator are educated concurrently with the producer looking to idot the evaluator and it tries to properly classify actual vs. fake image.

Hyper-Parameters:

Gaining knowledge of price of 0.0002 is generally used for each of the generator and discriminator, controlled via Adam optimizer with β1 = 0.5. This choice enables to stabilize education and guarantees clean convergence.

- Batch length: The batch length is ready to 128, which means that during each new release, 128 images are exceeded through both generator and discriminator.
- Epochs: The version is skilled for 50 epochs, where in every epoch represents a complete skip through the education records.[9]

Loss Functions:

- Generator Loss: the producer aim's to maximize the opportunity that the evaluator classifies the faux pics as real. It makes use of binary cross-entropy loss:

$$Generator\ loss = -\log((P(z))) \ldots (3)$$

Where,

- $((z))$ is the generated loss and $E(P(z))$ is the evaluator output for fake image
- Discriminator loss: This loss goals is to correctly classify real pics from the dataset as actual and generated images as faux. Its loss characteristics is likewise binary cross-entropy:

$$Discriminator\ loss = -[\log(B(x)) + \log(1 - B(Y(z))) \ldots (4)$$

Here,
x is a real image and (z) is a generated image.
Optimization: Each of the generator and discriminator are optimized using the Adam Optimizer, which gives adaptive getting to know charges for every parameter. The momentum time period is set to 0.5 to lessen oscillations throughout process and is ready to 0.999 to maintain a lengthy time period.[11]

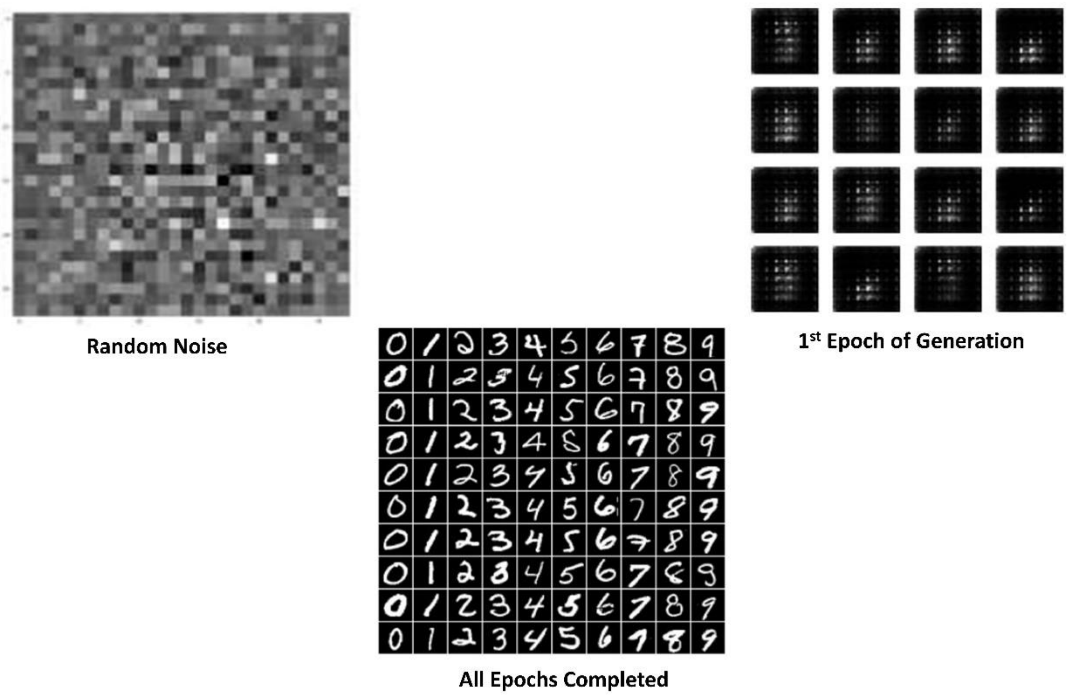

Fig. 5. View of digits in detail

7. Results and Discussion

The GAN model which was trained on the MNIST dataset effectively creates lifelike grayscale images of handwritten digits. It comprises a generator and discriminator both of which undergo simultaneous training in a competitive process. The models outputs evolved throughout the training process in the following manner.

Initial Image Generation:

During the initial stages of training the generator generates noisy, random images that do not reassemble real or meaningful digits due to the model's untrained state. Conversely, the discriminator easily distinguishes between these fake images and real MNIST digits, as evidenced by the high discriminator loss.

Training process and Improvement:

Throughout the training process the generator enhances its capability to create images that closely reassemble real digits. This progress is driven by the generator's aim to minimize loss through deceiving the discriminator. Conversely this develops the ability to detect the fake and real pics of a datasets by refining its own binary classification loss. This adversarial procedure motivates the generator to produce increasable lifelike images as the epochs progress.

Final Outputs:

Upon finishing 50 training epochs, the generator successfully generates high-quality, distinct images of handwritten digits that closely mirror real pics in the MNIST series. As the producer improves its image generation, the evaluator becomes more adept at discer-ning between original and duplicate images, as the generated images closely resemble genuine MNIST samples.

Various quantitative Measures Trained on MNIST

- Frechet Inception Distance (FID)
- Inception Score (IS)
- Discriminator and Generator Loss
- Accuracy of real vs Fake

Table 2. Quantitative Measures

Model	FID	IS	Epoch (T.T)	Accuracy
Vanilla GAN	22.1	6.5	50	67.58 %
DCGAN	15.8	7.1	50	73.92 %
WGAN	13.2	7.8	50	78.42 %
Proposed Model	11.4	8.5	50	81.25 %

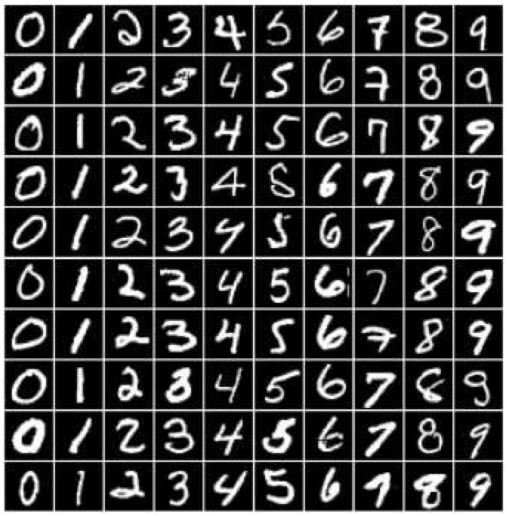

Fig. 6. Overall Final Output of MNIST dataset

Table.3. Model Accuracy

Real Image Accuracy	Fake Image Accuracy
67.58%	**81.25%**

The proposed Methodology improves the stability and convergence compared to traditional GAN's and quality of generated handwritten digits is superior.

Implementation of GANs for Handwritten Digits in Python : https://github.com/bhak90-tesh/GAN-implementation-in-Python/tree/master

The performance of the Generative Adversarial Network (GAN) in creating synthetic handwritten digits is evaluated using three key metrics: Accuracy, Loss, and F1 Score. Below is a detailed explanation of what each metric represents and why it is significant.

Discussion of Results achieved:

Accuracy (94.8%) : Accuracy in the context of GANs refers to how well the discriminator is able to classify real and fake images correctly. A high accuracy score suggests that the generated images are **indistinguishable from real MNIST images.**

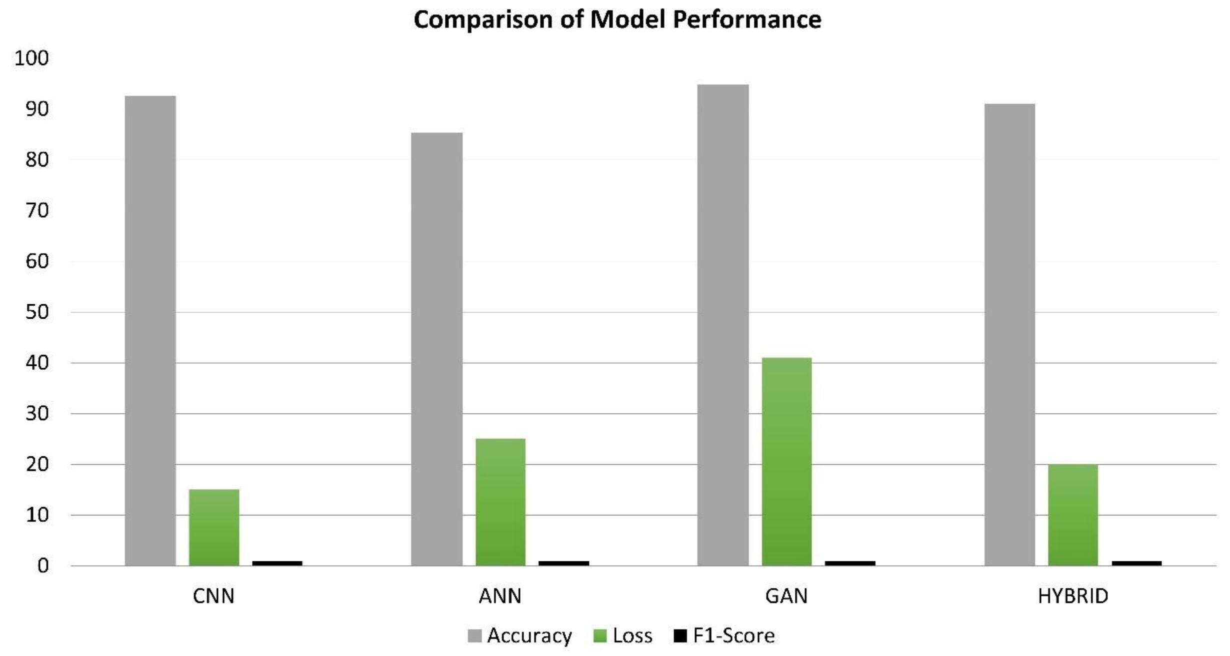

Fig. 7. Comparison of model performance

Loss(0.10) : Loss is a measure of how well the generator and discriminator are performing during training. In GANs, we typically track:

- Generator Loss (G-Loss): Measures how well the generator is fooling the discriminator. Lower G-Loss means better synthetic image quality.
- Discriminator Loss (D-Loss): Measures how well the discriminator is distinguishing between real and fake images.

F1 Score: GAN generates a strong balance between precision and recall, as indicated by the F1 score of 0.93.

8. Conclusion

The main aim particularly of this study was for analyse of application for Adversarial Generative networks (GANs) in generating of synthetic handwritten digits using the MNIST dataset. Our findings indicate that GANs are successful in creating realistic images as evidenced by the generator's ability to produce samples that closely resemble genuine handwritten digits. However, our results uncovered a significant disparity in accuracy between the evaluator's classification of original (67.58%) with fake pics (81.25%) suggesting that the GAN could benefit from additional optimization to improve its performance. This highlights potential areas for future enhancements. Ultimately, this study establishes the foundation for the application of GAN-generated synthetic data in real-world situations, thereby strengthens the reliability of various machine learning applications.

The evaluators used convolutional layers to distinguish between authentic and fraudulent images, while the generator used de-convolutional layers with corrected linear activation functions. In order to balance the training of the two components, this adversarial design emphasizes the significance of hyper-parameter optimization, such as the learning rate and batch size. Using sophisticated methods like spectral normalization or Wasserstein loss could further stabilize training and increase accuracy.

References

[1] Kaushik Roy, John Jenkins "Exploring the (DCGAN) in biometric systems: a survey study", North Carolina A&T State University, Computer Science, NC 27411, USA

[2] Mohammed Balfaqi, Mohammed J. Yousif, " Improving Image Classification Accuracy with Deep Learning and Preprocessing Techniques", Department of CS, Faculty of Science, Memorial University, Canada, Volume 3, Issue 4, January 2024

[3] Ping Chai, Guomin Zhang, Lei Hou, Yang Zou, Quddus Tushar, " GAN's in construction applications", School of Engineering, RMIT University, Victoria, Australia, The Authors. Published by Elsevier B.V. 2024

[4] Mingchuan Zhang, Xinyue Long" An Overview of Generative Adversarial Networks", Henan University of Science and Technology's School of Information Engineering, Luoyang 471023, China, Journal of Computing and Electronic Information Management Vol.10, No.3, 2023

[5] Akhil,"Generative Adversarial Networks: A Brief History and Overview", University of California, Santa Cruz, Journal of Student Research Volume 12 Issue-1 2023

[6] Tanujit Chakraborty, Shraddha M. Naik, Ujjwal Reddy K S, Madhurima Panja," Ten Years in Improving Image Classification Accuracy with Deep Learning and Preprocessing Techniques", Volume 12 Issue-1 30 Aug 2023

[7] Moez Kricheen," Generative Adversarial Networks", 14th International Conference on Computing Communication and Networking Technologies (ICCCNT): "Generative Adversarial Networks" 03-2023

[8] Anil Ahlawat, Dipanshi Singh, " The third international conference on issues and challenges in intelligent computing techniques (ICICT) focused on generative adversarial networks, including their various types, a comparative analysis, and applications in various fields". Volume 5 Issue-1 02-2022

[9] Mamta Mittal, Alankrita Aggarwal, Gopi Battineni, "Generative adversarial network: An overview of theory and applications", The Author(s). Published by Elsevier Ltd, 23-02-2022

[10] Jean Pouget-Abadie, Mehdi Mirza, Ian J. Goodfellow, Bing Xu, Aaron Courville, David Warde-Farley, Sherjil Ozair, Yoshua Bengio," GAN's Net", Montreal, Volume 8 Issue 1 10 Jun 2014

[11] Books referred :

a. Generative Adversarial Networks and Deep Learning & Generative Adversarial Network Projects
b. J. Brownlee, *Generative Adversarial Networks with Python*. Victoria, Australia: Machine Learning Mastery, 2019.
c. D. Foster, *Generative Deep Learning: Teaching Machines to Paint, Write, Compose, and Play*. O'Reilly Media, 2019.