# Generation of Synthetic Data Using GANs

Shashwat Mishra
Rashi Mohta
Sahil Gupta
Harit Gupta

June 13, 2024

# ABSTRACT

Data scarcity and quality pose substantial challenges in the development and deployment of AI models, particularly in the finance industry, where accurate and reliable data is paramount. Traditional methods of data augmentation and synthesis often fall short in maintaining the intricate properties and relationships inherent in real-world data. Generative Adversarial Networks (GANs) emerge as a powerful solution, offering the capability to generate high-quality synthetic data that closely mirrors real data, while also preserving privacy by ensuring that the synthetic data cannot be traced back to any individual in the real dataset.

This white paper delves into the application of GANs for generating synthetic data, focusing on two advanced models: Conditional Tabular GAN (CTGAN) for tabular data and Time GAN for time series data. CTGAN is particularly adept at handling the complexities of tabular data, capturing the dependencies between features to produce realistic synthetic datasets. Time GAN, on the other hand, is designed to generate synthetic time series data that maintains the temporal dynamics and correlations present in the original data.

In our study, we apply CTGAN to the problem of generating synthetic datasets for credit card fraud detection. This domain presents significant challenges due to the imbalanced nature of the data and the critical importance of preserving feature relationships to accurately model fraudulent behavior. Our results demonstrate that CTGAN can produce synthetic datasets that not only enhance model training but also improve the robustness and generalization of fraud detection algorithms.

Furthermore, we explore the use of Time GAN for generating synthetic time series data, with a specific focus on stock price prediction for Google. The ability to generate realistic time series data is crucial for developing and testing predictive models in finance, where historical data drives forecasting accuracy. Our experiments show that Time GAN effectively replicates the patterns and trends in Google stock prices, providing a valuable tool for financial analysts and AI model developers.

# Contents



# List of Figures

# 1 Introduction

Data is the cornerstone of AI development, yet it often presents the most significant bottleneck. Collecting data can be prohibitively expensive and time-consuming, especially when dealing with rare events or phenomena. For instance, financial markets experience rare extreme events that are difficult to capture and costly to simulate. Moreover, data processing is prone to human errors, leading to potential biases and inaccuracies.

## 1.1 Challenges in Data Collection

Data collection faces several challenges:

- **Rarity of Events**: Certain events, such as financial crises or rare diseases, occur infrequently, making data collection difficult and expensive.
- **Cost**: Gathering extensive datasets can be cost-prohibitive, involving significant resources and time.
- **Human Error**: Manual data entry and processing are prone to errors, which can introduce biases and reduce data quality.

## 1.2 Advantages of Synthetic Data

Synthetic data can mitigate these issues by providing a cost-effective and scalable alternative. It enables the creation of large datasets that mirror the statistical properties of real data without the associated collection challenges.

Advantages include:

- **Cost Efficiency**: Synthetic data eliminates the need for expensive data collection processes.
- **Privacy**: It allows for the generation of data that does not contain sensitive information, ensuring privacy compliance.
- **Accessibility**: Researchers and developers can access diverse datasets without restrictions.
- **Augmentation**: Synthetic data can augment real datasets, enhancing the training of AI models.

# 2 Theory of Generative Adversarial Networks (GANs)

GANs represent a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in 2014. The fundamental concept behind GANs is the adversarial process, where two neural networks, the generator and the discrim- inator, contest with each other in a game-theoretic scenario

## 2.1 The Architecture of GANs

GANs, introduced by Goodfellow et al. (2014), At the core of GANs lie two main components:

- **Generator**: This neural network is responsible for producing fake data samples. Its goal is to generate data that is so convincing that the discriminator cannot reliably tell whether it is fake or real.
- **Discriminator**: This neural network acts as a critic that examines the data and attempts to distinguish between real and synthetic samples. Its objective is to correctly identify the authenticity of the data presented to it.

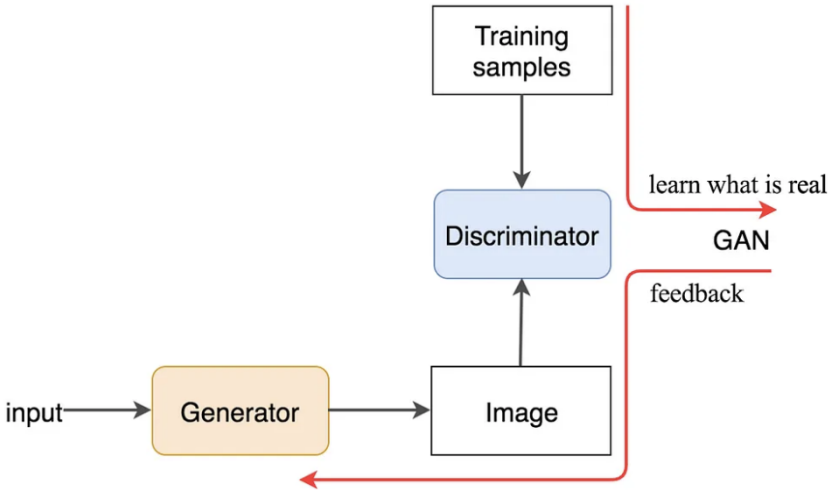



Figure 1: Generator and Discriminator

The goal is for the generator to produce outputs that the discriminator classifies as indistinguishable from real data, i.e. D(G(z))=1. This is achieved by propagating the objective through the generator and updating its parameters accordingly, encouraging it to generate data that the discriminator perceives as authentic.

## 2.2 Training Process of GANs

Training of both networks occurs in an alternating fashion, fostering a dynamic adversarial process. The discriminator continually refines its ability to discern subtle differences between real and synthetic data, while the generator simultaneously improves its capability to produce high-quality synthetic data that can deceive the discriminator.

This iterative competition drives both networks to enhance their performance, ultimately leading to a model that generates data indistinguishable from the real data. The GAN model converges when the discriminator can no longer reliably differentiate between real and synthetic data, resulting in the generation of highly realistic data.

Mathematically, GANs are formulated through the following minimax game where G wants to minimize V while D wants to maximize it:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x)] + E_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{1}$$

where:

- $G$ is the generator network.
- $D$ is the discriminator network.
- $x$ is the real data sample from the distribution $p_{data}$.
- $z$ is the noise variable from the distribution $p_z$.

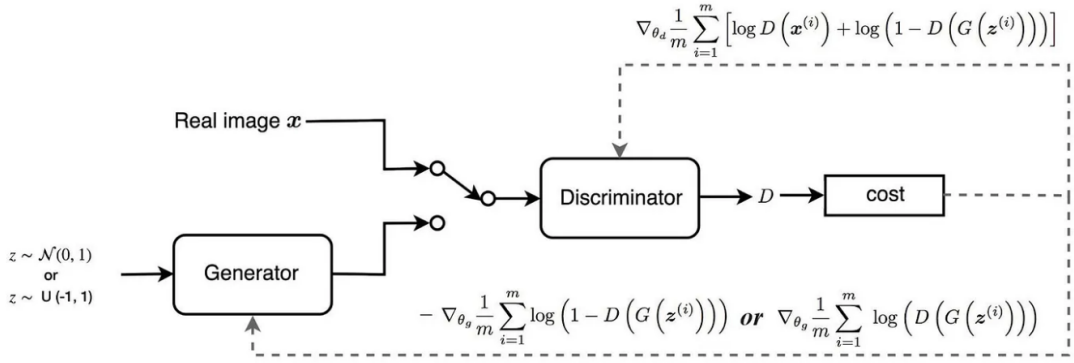



Figure 2: Data flow and Gradients for Generator and Discriminator

Once both objective functions are defined, they are learned jointly by the alternating gradient descent algorithm. We fix the generator model's parameters and perform a single iteration of gradient ascend on the discriminator using the real and the generated images. Then we switch sides. Fix the discriminator and train the generator for another single iteration. We train both networks in alternating steps until the model converges.

The pseudo-code below puts everything together and shows how GAN is trained.

## 2.3 Conditional Tabular GAN (CTGAN)

Conditional Tabular GAN (CTGAN) is an advanced type of Generative Adversarial Network specifically designed to generate realistic synthetic tabular data. Unlike conventional GANs, which are typically used for generating continuous data such as images. CTGAN focuses on the complexities associated with tabular data, which often includes a mix of continuous and categorical variables, missing values, and imbalanced classes.

One of the primary distinctions between CTGAN and standard GANs lies in its conditioning mechanism. In a typical GAN, the generator creates data from a random noise vector, which the discriminator then evaluates. However, CTGAN introduces conditional vectors into the generator, enabling it to produce data that adheres to specific conditions or distributions present in the real dataset. This conditioning is particularly useful for tabular data, where the relationships between variables can be intricate and highly structured.

**Algorithm 1** Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

**for** number of training iterations **do**

**for** $k$ steps **do**

- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Sample minibatch of $m$ examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
- Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[ \log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

**end for**

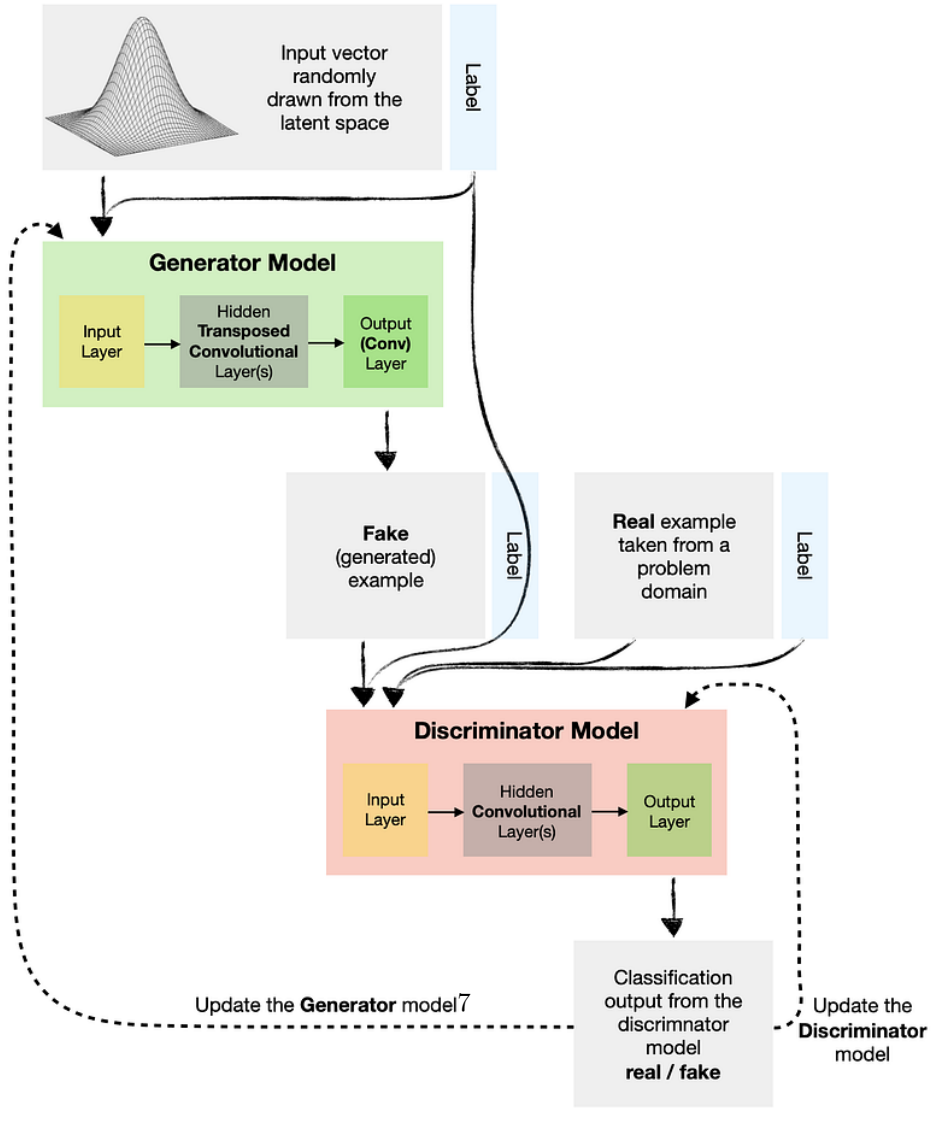
- Sample minibatch of $m$ noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
- Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

**end for**

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

Figure 3: Pseducode for Back Propagation



## 2.4 Training Process of cGANs

CTGAN uses mode-specific normalization and training-by-sampling techniques to improve the generation process. The CTGAN model optimizes the following conditional objective

$$\min_G \max_D L(D,G) = E_{x\sim p_{\text{data}}(x)}[\log D(x \mid c)] + E_{z\sim p_z(z)}[\log(1 - D(G(z \mid c)))] \tag{2}$$

where:

- D( x | c ) is the probability that $x$ is a real sample given the condition $c$
- G( z | c ) is the generator's output when generating a sample given the noise vector $z$ and the condition $c$.

**Algorithm 1** Minibatch Stochastic Gradient Descent Training of CTGAN

The number of steps to apply to the discriminator, $k$, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.
**for** number of training iterations **do**
  **for** $k$ steps **do**
    Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ and their conditions $\{c^{(1)}, \ldots, c^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
    Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)} \mid c^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)} \mid c^{(i)}\right)\right)\right)\right]$$

  **end for**
  Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
  Sample conditions $\{c^{(1)}, \ldots, c^{(m)}\}$ from condition distribution $p(c)$.
  Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)} \mid c^{(i)}\right)\right)\right)$$

**end for**
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments. =0

## 2.5 Time-series Generative Adversarial Networks (TimeGAN)

Time-series data is prevalent in various domains such as finance, healthcare, and sensor networks. Generating realistic synthetic time-series data has numerous applications, including data augmentation for machine learning models, privacy-preserving data sharing, and anomaly detection. Traditional generative models often struggle with the temporal dependencies inherent in time-series data. In 2019, Time-series Generative Adversarial Networks (TimeGAN) was proposed to address this challenge by generating realistic time-series data while preserving the temporal dynamics.

TimeGAN introduces a novel framework that extends the capabilities of Generative Adversarial Networks (GANs) to handle the unique characteristics of time-series data. Unlike conventional GAN architectures, TimeGAN incorporates both unsupervised and supervised learning techniques to capture the temporal dependencies and conditional distributions within the data.

## 2.6 Architecture of TimeGAN

The TimeGAN architecture consists of four main components:

- the generator

- the discriminator
- the embedding network
- the recovery network

The generator aims to create synthetic time-series data that mimics the real data distribution. The discriminator, on the other hand, evaluates the authenticity of the generated data by distinguishing it from real data.

One of the key innovations of TimeGAN is the introduction of the embedding network. This network is responsible for reducing the dimensionality of the adversarial learning space, thereby simplifying the learning process. The embedding network encodes the time-series data into a latent space, capturing the essential temporal features.

Another significant contribution of TimeGAN is the incorporation of supervised loss. This loss encourages the model to capture the time-conditional distributions within the data by using the original data as supervision. This is different from traditional GANs, which typically rely solely on unsupervised adversarial loss. By combining supervised and unsupervised learning, TimeGAN ensures that the generated time-series data not only looks realistic but also preserves the temporal dynamics and dependencies of the original data.

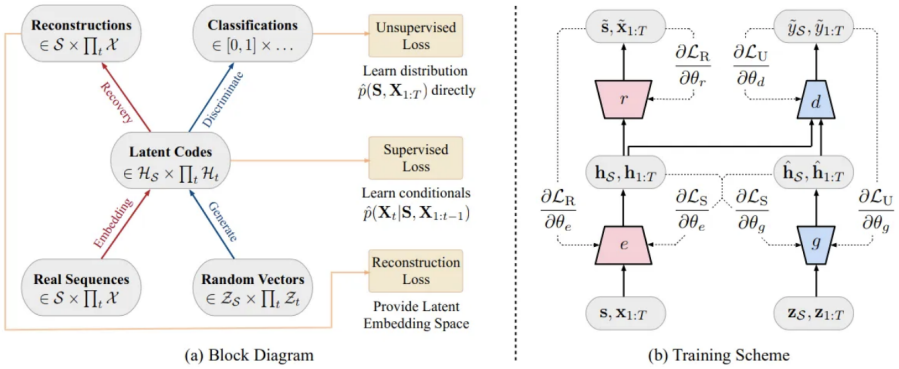



Figure 5: TimeGAN Architecture

## 2.7 Training TimeGAN

Training TimeGAN involves optimizing the generator and discriminator through adversarial learning, similar to other GAN architectures. However, TimeGAN also integrates a supervised component into the training process. The supervised loss guides the generator to produce time-series data that follows the conditional distribution of the real data. This dual objective helps TimeGAN capture both the global distribution and the local temporal dependencies of the time-series data.

Time-series Generative Adversarial Networks (TimeGAN) employs a hybrid loss function to effectively generate realistic time-series data while preserving temporal dependencies. The total loss function in TimeGAN comprises four main components: adversarial loss, supervised loss, embedding loss, and recovery loss.

### Adversarial Loss

The adversarial loss encourages the generator to produce data that the discriminator cannot distinguish from real data. This loss is akin to the traditional GAN adversarial loss.

- **Discriminator Loss**

$$L_D = -E_{x \sim p_{data}(x)}[\log D(x)] - E_{z \sim p_z(z)}[\log(1 - D(G(z)))] \tag{3}$$

- **Generator Loss**

$$L_G = -E_{z \sim p_z(z)}[\log D(G(z))] \tag{4}$$

**Supervised Loss**

The supervised loss ensures that the generator captures the temporal dynamics by using the original time-series data as supervision. This loss is calculated based on the real and generated sequences.

$$L_{supervised} = E_{x \sim p_{data}(x)}[\|h(x) - \hat{h}(x)\|^2] \tag{5}$$

where $h(x)$ and $\hat{h}(x)$ represent the embeddings of the real and generated sequences, respectively.

**Embedding Loss**

The embedding loss helps preserve the temporal structure by accurately mapping real data to the latent space.

$$L_{embedding} = E_{x \sim p_{data}(x)}[\|x - \hat{x}\|^2] \tag{6}$$

where $\hat{x}$ is the recovered sequence from the latent space.

**Recovery Loss**

The recovery loss ensures that the sequence recovered from the latent space matches the original sequence, maintaining the temporal relationships.

$$L_{recovery} = E_{x \sim p_{data}(x)}[\|x - R(E(x))\|^2] \tag{7}$$

where $R$ is the recovery network and $E$ is the embedding network.

**Total Loss**

The total loss for the TimeGAN model is a combination of these individual loss components:

$$L_{total} = \alpha L_D + \beta L_G + \gamma L_{supervised} + \delta L_{embedding} + \eta L_{recovery} \tag{8}$$

Here, $\alpha, \beta, \gamma, \delta$, and $\eta$ are hyperparameters that control the contribution of each loss component to the total loss.

**Algorithm 2** TimeGAN Training Algorithm

**Input:** Training data $\{x^{(1)}, x^{(2)}, \ldots, x^{(N)}\}$
**Output:** Trained TimeGAN model
Initialize the parameters of the embedding network $E$, generator $G$, discriminator $D$, and recovery network $R$
Define the hyperparameters: learning rates, batch size, number of iterations
**for** number of training iterations **do**
  **1. Embedding and Recovery Training:**
  **for** each batch of real data samples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ **do**
    Encode real data to latent space: $h = E(x)$
    Recover data from latent space: $\hat{x} = R(h)$
    Calculate embedding loss: $L_{embedding} = \frac{1}{m} \sum_{i=1}^{m} \|x^{(i)} - \hat{x}^{(i)}\|^2$
    Update parameters of $E$ and $R$ using gradient descent to minimize $L_{embedding}$
  **end for**
  **2. Adversarial Training:**
  **for** each batch of noise samples $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ and real data samples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ **do**
    Generate synthetic data: $\hat{x} = G(z)$
    Calculate discriminator loss on real data: $L_{D,real} = -\frac{1}{m} \sum_{i=1}^{m} \log D(x^{(i)})$
    Calculate discriminator loss on synthetic data: $L_{D,fake} = -\frac{1}{m} \sum_{i=1}^{m} \log(1 - D(\hat{x}^{(i)}))$
    Total discriminator loss: $L_D = L_{D,real} + L_{D,fake}$
    Update parameters of $D$ using gradient descent to minimize $L_D$
  **end for**
  **for** each batch of noise samples $\{z^{(1)}, z^{(2)}, \ldots, z^{(m)}\}$ **do**
    Generate synthetic data: $\hat{x} = G(z)$
    Calculate generator loss: $L_G = -\frac{1}{m} \sum_{i=1}^{m} \log D(\hat{x}^{(i)})$
    Update parameters of $G$ using gradient descent to minimize $L_G$
  **end for**
  **3. Supervised Training:**
  **for** each batch of real data samples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ **do**
    Encode real data to latent space: $h = E(x)$
    Generate synthetic data from latent space: $\hat{x} = G(h)$
    Calculate supervised loss: $L_{supervised} = \frac{1}{m} \sum_{i=1}^{m} \|h - \hat{h}\|^2$
    Update parameters of $E$ and $G$ using gradient descent to minimize $L_{supervised}$
  **end for**
  **4. Recovery Training:**
  **for** each batch of real data samples $\{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$ **do**
    Encode real data to latent space: $h = E(x)$
    Recover data from latent space: $\hat{x} = R(h)$
    Calculate recovery loss: $L_{recovery} = \frac{1}{m} \sum_{i=1}^{m} \|x - R(h)\|^2$
    Update parameters of $E$ and $R$ using gradient descent to minimize $L_{recovery}$
  **end for**
**end for**
**Return:** Trained TimeGAN model =0

Figure 6: TimeGAN PseduoCode

# 3 Testing and Results

## 3.1 Overview

In this section we present the results of our experiment on generating synthetic data using Generative Adversarial Networks (GANs) for two of our use cases as follows:

1. Credit card transactions
2. Adult census data

We evaluate the performance of our model based on both qualitative and quantitative metrics, comparing the synthetic data with the real dataset.We also see the magnitute of variations that our synthetic and real data can produce.

## 3.2 Feature distribution comparision

1. **Credit card transactions**:Figure 7 visualises the transactions in both the real dataset and the synthetic dataset. We see that the real dataset is sparse, and may not be enough. We enhance the quantity with our synthetic data, which follows a similar distribution, and which can easily be used for training complex data-hungry machine learning models.

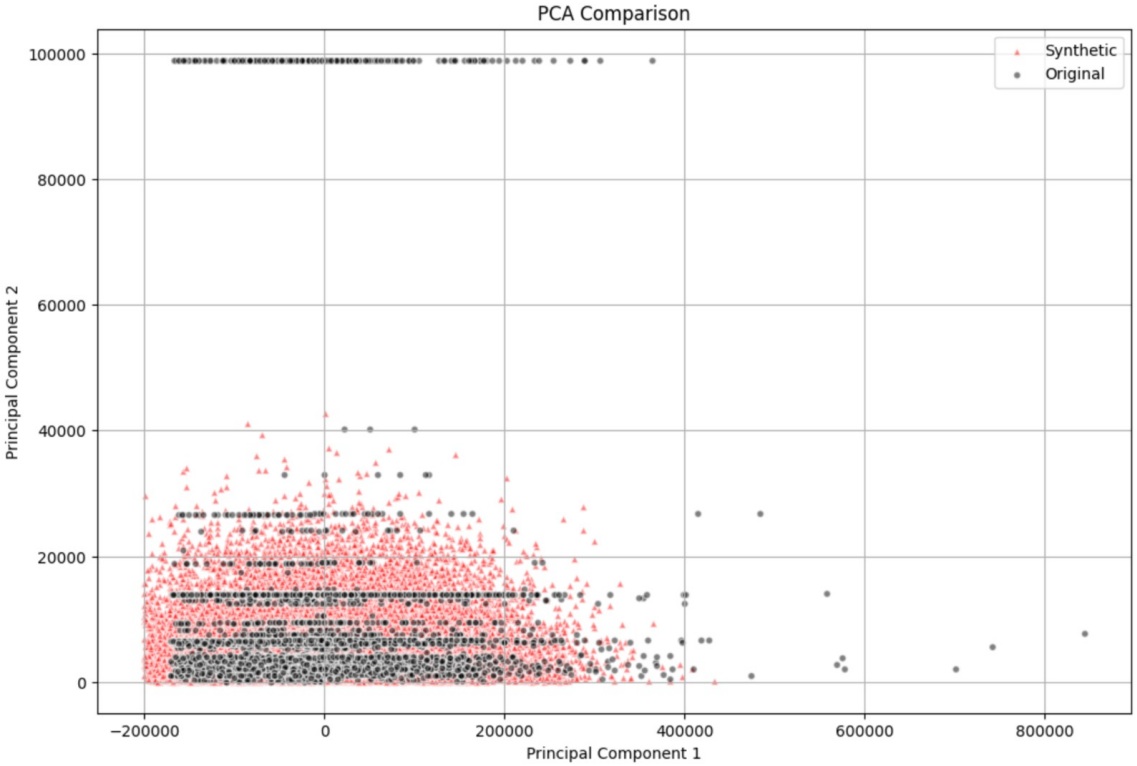



Figure 7: Credit card transactions

2. **Adult census data**: In this case, we first check in Figure 8 the various covariances that we observe in the input data, on the basis of which our output data is generated.

| | age | workclass | fnlwgt | education | education num | marital status | occupation | relationship | race | sex | capital gain | capital loss | hours per week | native country |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| age | 0.000 | 2.329 | 1.030 | 2.035 | 1.855 | 1.340 | 0.345 | 1.211 | 1.700 | 0.971 | 0.950 | 0.958 | 0.839 | 0.373 |
| workclass | 2.329 | 0.000 | 0.511 | 0.510 | 0.302 | 0.784 | 0.173 | 1.066 | 0.666 | 0.869 | 0.480 | 1.697 | 0.432 | 0.269 |
| fnlwgt | 1.030 | 0.511 | 0.000 | 0.356 | 0.201 | 0.968 | 6.439 | 1.582 | 1.968 | 0.745 | 0.351 | 0.030 | 1.666 | 0.473 |
| education | 2.035 | 0.510 | 0.356 | 0.000 | 0.226 | 0.015 | 1.101 | 0.507 | 0.717 | 0.226 | 0.383 | 0.177 | 0.190 | 0.174 |
| education-num | 1.855 | 0.302 | 0.201 | 0.226 | 0.000 | 0.463 | 0.398 | 1.008 | 1.038 | 4.578 | 0.375 | 0.414 | 0.341 | 0.133 |
| marital-status | 1.340 | 0.784 | 0.968 | 0.015 | 0.463 | 0.000 | 0.478 | 0.695 | 1.008 | 0.831 | 0.869 | 0.878 | 0.936 | 0.038 |
| occupation | 0.345 | 0.173 | 6.439 | 1.101 | 0.398 | 0.478 | 0.000 | 0.687 | 1.529 | 0.519 | 0.596 | 0.736 | 0.296 | 0.137 |
| relationship | 1.211 | 1.066 | 1.582 | 0.507 | 1.008 | 0.695 | 0.687 | 0.000 | 1.167 | 1.508 | 1.098 | 0.969 | 1.101 | 1.313 |
| race | 1.700 | 0.666 | 1.968 | 0.717 | 1.038 | 1.008 | 1.529 | 1.167 | 0.000 | 1.131 | 1.293 | 1.290 | 1.140 | 0.105 |
| sex | 0.971 | 0.869 | 0.745 | 0.226 | 4.578 | 0.831 | 0.519 | 1.508 | 1.131 | 0.000 | 0.986 | 0.805 | 0.607 | 0.227 |
| capital-gain | 0.950 | 0.480 | 0.351 | 0.383 | 0.375 | 0.869 | 0.596 | 1.098 | 1.293 | 0.986 | 0.000 | 1.071 | 1.026 | 0.526 |
| capital-loss | 0.958 | 1.697 | 0.030 | 0.177 | 0.414 | 0.878 | 0.736 | 0.969 | 1.290 | 0.805 | 1.071 | 0.000 | 0.921 | 0.781 |
| hours-per-week | 0.839 | 0.432 | 1.666 | 0.190 | 0.341 | 0.936 | 0.296 | 1.101 | 1.140 | 0.607 | 1.026 | 0.921 | 0.000 | 4.293 |
| native-country | 0.373 | 0.269 | 0.473 | 0.174 | 0.133 | 0.038 | 0.137 | 1.313 | 0.105 | 0.227 | 0.526 | 0.781 | 4.293 | 0.000 |

Figure 8: Covariance between features

After generation of synthetic data, we observe the mean and standard deviation(STD) differences

between mean and standard deviation of our real and synthetic data. These can be observed below:

| | Mean difference(%) | STD difference(%) |
|---|---|---|
| age | 0.000121 | 0.003302 |
| workclass | 0.006623 | 0.085787 |
| fnlwgt | 0.000913 | 0.000116 |
| education | 0.086580 | 0.182109 |
| education-num | 0.024354 | 0.249180 |
| marital-status | 0.041328 | 0.005342 |
| occupation | 0.037125 | 0.022192 |
| relationship | 0.000057 | 0.000357 |
| sex | 0.024593 | 0.199508 |
| capital-gain | 0.042823 | 0.002868 |
| capital-loss | 0.004981 | 0.000064 |
| hours-per-week | 0.000187 | 0.002652 |
| native-country | 0.050303 | 0.549395 |
| target | 0.014668 | 0.016460 |

Here we observe that the percentage-wise difference between the real and synthetic data is minimal, which shows that the synthetic data replicated the patters of the real underlying datavery well, and can be used for future applications.

## 3.3 Discussion

Our results indicate that the GAN-generated synthetic data closely resembles the real data in terms of statistical properties and feature distributions in both credit card transaction and Adult census data. The qualitative analysis further confirms the diversity and realism of the synthetic data, suggesting its potential utility for training fraud detection models. However, we also note some limitations, such as the need for further refinement to address specific edge cases or biases present in the real dataset.

## 3.4 Conclusion

In this study, we have demonstrated the effectiveness of using GANs to generate synthetic data for multiple use cases. Our results suggest that synthetic data can serve as a valuable resource for augmenting limited datasets and improving the robustness of machine learning models, particularly in the domain of fraud detection. In future research, we aim to focus on optimizing the GAN architecture, and investigating the generalization capabilities of synthetic data across different datasets and use cases thus improving availability of synthetic data in all required cases.