# GAN-based Synthetic Data Generation: A Comprehensive Report

Rahul Madhogarhia (CS24MTECH12004)
Ankush Chhabra (CS24MTECH14004)
Chalasani Vineeth (CS24MTECH11023)

April 22, 2025

## Abstract

This report documents the process and results of generating synthetic data using a Generative Adversarial Network (GAN). The approach involves data preprocessing with a MinMax scaler, training a GAN composed of a Generator and a Critic using the Wasserstein GAN framework with a gradient penalty, and subsequently evaluating the quality of the synthetic data through visual and statistical analyses including t-SNE plots, density estimation via Kernel Density Estimate(KDE) plots, and correlation heatmaps.

## 1 Problem Statement

The objective of this project is to generate synthetic data that closely mimics the characteristics of a given real-world dataset. The main goals are:

- Preprocess the dataset using normalization techniques.
- Develop a GAN framework (Generator and Critic) to model the underlying distribution of the original data.
- Train the GAN using a gradient penalty term to ensure stability.
- Generate a set of synthetic data and evaluate its quality by comparing it with the original dataset.

## 2 Dataset Description

The dataset used in this study is stored in an Excel file `data.xlsx`. It consists of numerical features which are scaled using a MinMax normalization before being fed into the GAN. This ensures all features are in a similar range, which is beneficial for model training and convergence.

## 3 Methodology

### 3.1 Preprocessing and Data Loader

The real data is imported from an Excel file and all numerical columns are normalized using the MinMaxScaler. The normalized data is converted to a PyTorch tensor and loaded using a `DataLoader` for efficient minibatch processing.

### 3.2 GAN Model Architecture

The GAN model consists of:

- **Generator:** A neural network that maps random noise vectors to data points that mimic the original data distribution.
- **Critic:** A neural network (used instead of a discriminator in a Wasserstein GAN) that assesses the quality of the generated samples compared to the real data.

### 3.3 Training Procedure and Pseudocode

The GAN is trained under the Wasserstein loss formulation, using a gradient penalty to enforce the Lipschitz constraint on the Critic. The training loop consists of:

1. For each batch of real data:
   - 1.1. **Critic Training:** Repeat for `n_critic` steps.
     - 1.1.1. Generate fake data samples using the Generator.
     - 1.1.2. Compute the critic scores for both real and fake samples.
     - 1.1.3. Evaluate the gradient penalty on interpolated samples.
     - 1.1.4. Calculate the Critic loss using the Wasserstein distance augmented by the gradient penalty.
     - 1.1.5. Update the Critic's parameters.
   - 1.2. **Generator Training:**
     - 1.2.1. Generate a new batch of fake data.
     - 1.2.2. Compute the generator loss as the negative mean of the Critic's score on the generated data.
     - 1.2.3. Update the Generator's parameters.
2. Log progress periodically.
3. Generate synthetic samples after training (using inverse scaling) after training.

**Algorithm 1** GAN Training Algorithm

```
1: **Input:** Real dataset $X$, number of epochs, batch size, learning rates, gradient penalty weight $\lambda_{gp}$.
2: for each epoch = 1 to epochs do
3:     for each batch $x_{\text{real}}$ in DataLoader do
4:         for $i = 1$ to $n_{\text{critic}}$ do
5:             Sample noise vector $z \sim \mathcal{N}(0, I)$.
6:             Generate fake data: $x_{\text{fake}} = G(z)$.
7:             Compute critic outputs: $C(x_{\text{real}})$, $C(x_{\text{fake}})$.
8:             Compute gradient penalty $gp = GP(C, x_{\text{real}}, x_{\text{fake}})$.
9:             Calculate loss: $L_C = -(C(x_{\text{real}}) - C(x_{\text{fake}})) + \lambda_{gp} \cdot gp$.
10:            Update Critic parameters by backpropagating $L_C$.
11:        end for
12:        Sample new noise vector $z$.
13:        Generate fake data: $x_{\text{fake}} = G(z)$.
14:        Compute generator loss: $L_G = -C(x_{\text{fake}})$.
15:        Update Generator parameters by backpropagating $L_G$.
16:    end for
17:    if epoch is a multiple of 100 then
18:        Log critic and generator losses.
19:    end if
20: end for
21: Output: Synthetic data generated by $G(z)$.
```

## 3.4 Training Progress

During training, the following log of Critic (C) and Generator (G) losses was recorded at intervals of 100 epochs:

```
Epoch [100/2000]  C Loss: -0.0136  G Loss: 0.5752
Epoch [200/2000]  C Loss: 0.0264  G Loss: -0.2646
Epoch [300/2000]  C Loss: 0.0091  G Loss: -0.0330
Epoch [400/2000]  C Loss: -0.0013  G Loss: -1.9928
Epoch [500/2000]  C Loss: 0.0367  G Loss: -0.0468
Epoch [600/2000]  C Loss: 0.0047  G Loss: 0.9010
Epoch [700/2000]  C Loss: 0.0094  G Loss: -0.5501
Epoch [800/2000]  C Loss: 0.0083  G Loss: -2.0629
Epoch [900/2000]  C Loss: 0.0408  G Loss: -2.5586
Epoch [1000/2000]  C Loss: -0.0091  G Loss: -0.8582
Epoch [1100/2000]  C Loss: -0.0305  G Loss: 3.3281
Epoch [1200/2000]  C Loss: 0.0224  G Loss: 4.6491
Epoch [1300/2000]  C Loss: 0.0211  G Loss: 2.1185
Epoch [1400/2000]  C Loss: -0.0038  G Loss: 0.4501
Epoch [1500/2000]  C Loss: -0.0033  G Loss: -1.7890
Epoch [1600/2000]  C Loss: 0.0030  G Loss: -2.5414
Epoch [1700/2000]  C Loss: 0.0275  G Loss: -2.9608
Epoch [1800/2000]  C Loss: -0.0450  G Loss: 0.0367
Epoch [1900/2000]  C Loss: -0.0208  G Loss: -0.9766
Epoch [2000/2000]  C Loss: -0.0083  G Loss: 1.5662
```

### 3.5 Synthetic Data Generation

Once the GAN is trained, synthetic data is generated by sampling 1199 noise vectors, passing them through the Generator, and then applying the inverse transformation of the MinMax scaler to obtain data in the original scale.

## 4 Post-Generation Analysis

### 4.1 t-SNE Visualization

- A 2-dimensional t-SNE projection was applied to both original and synthetic data samples.
- The results are plotted to visually compare the distributions in a reduced feature space.
- In the plot, the original and synthetic samples are clearly marked to assess overlap and similarity.

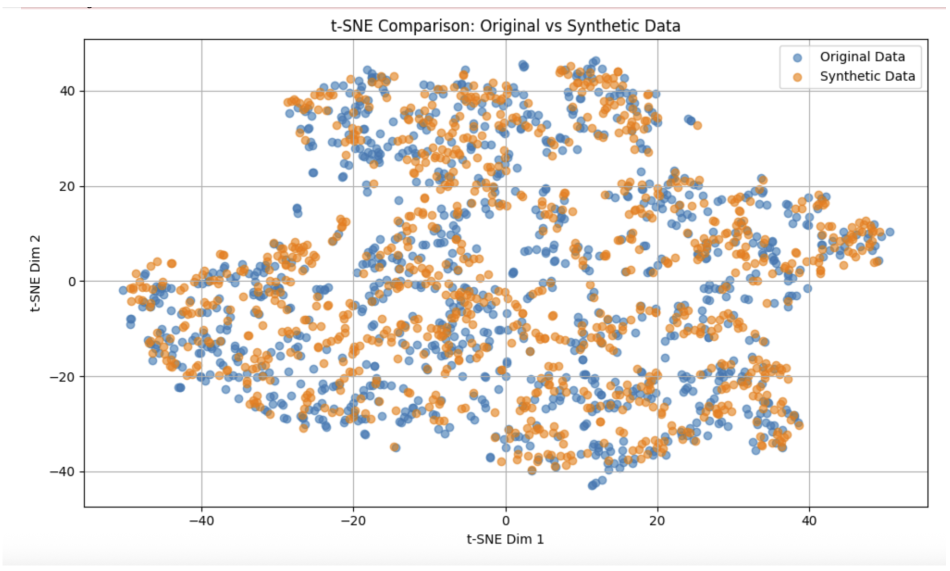


Figure 1: t-SNE projection comparing original (blue) and synthetic (orange) samples.

### 4.2 Distribution Comparison Using KDE

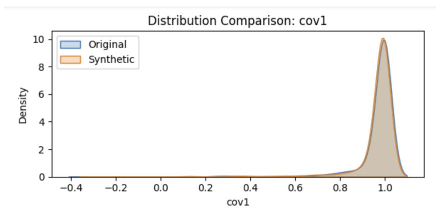
- For each numerical feature, KDE plots were generated.
- Separate KDE plots for the original data and synthetic data are presented in order to visually compare the distribution shapes.

### 4.3 Correlation Analysis

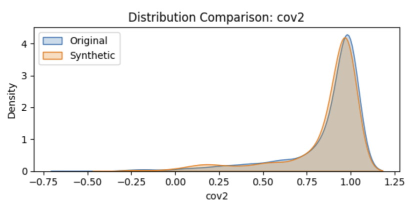
- Correlation matrices for both the real and synthetic datasets are computed.
- Two sets of heatmaps are plotted:
    1. Basic heatmaps without annotations for a quick visual inspection.
    2. Annotated heatmaps (with correlation coefficients) for detailed comparison.
- The correlation matrices are flattened and a similarity score is computed (as the Pearson correlation coefficient between the flattened arrays), expressed as a percentage.
- Correlation similarity score between real and synthetic data: 98.71%.

## 4.4 Density Distribution of Individual Features

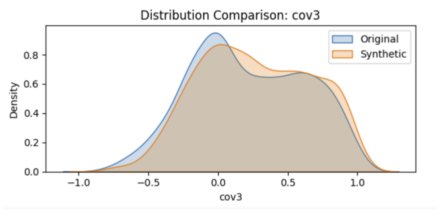
To provide a detailed comparison of how each individual feature is distributed in the original versus the synthetic dataset, Kernel Density Estimate (KDE) plots are shown below for all 10 features. Each plot displays the KDE curves for both the real and synthetic data.



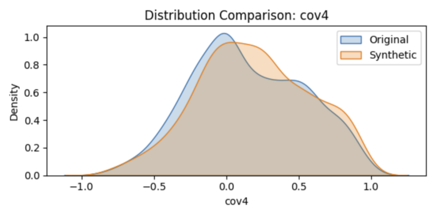
(a) Feature 0



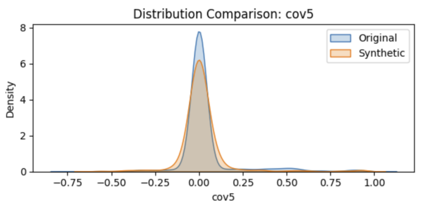
(b) Feature 1



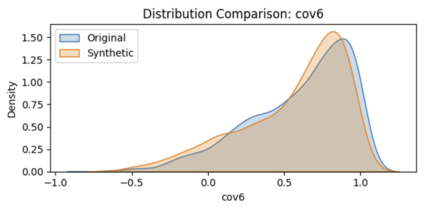
(c) Feature 2



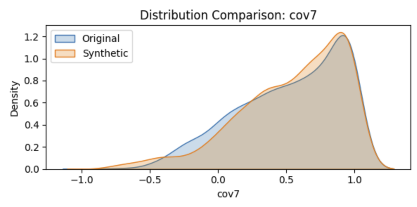
(d) Feature 3



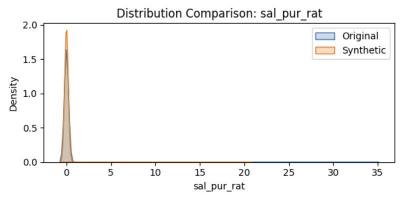
(e) Feature 4



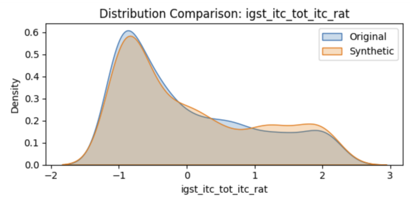
(f) Feature 5



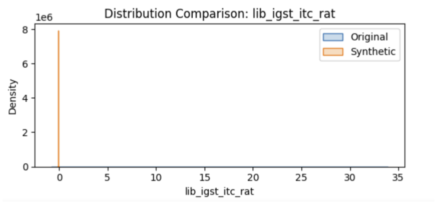
(g) Feature 6



(h) Feature 7



(i) Feature 8

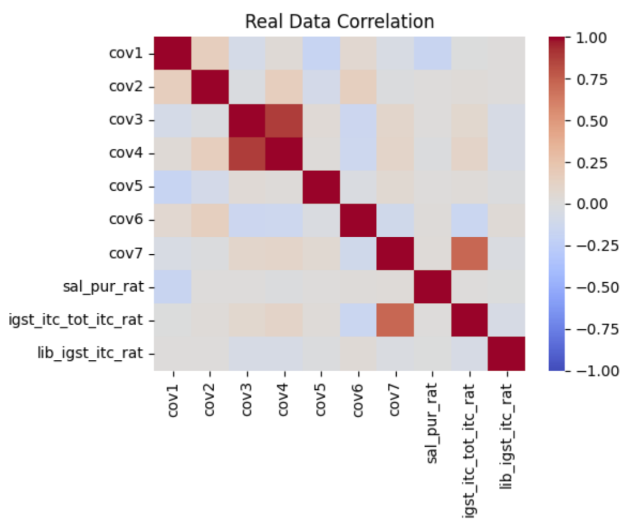


(j) Feature 9

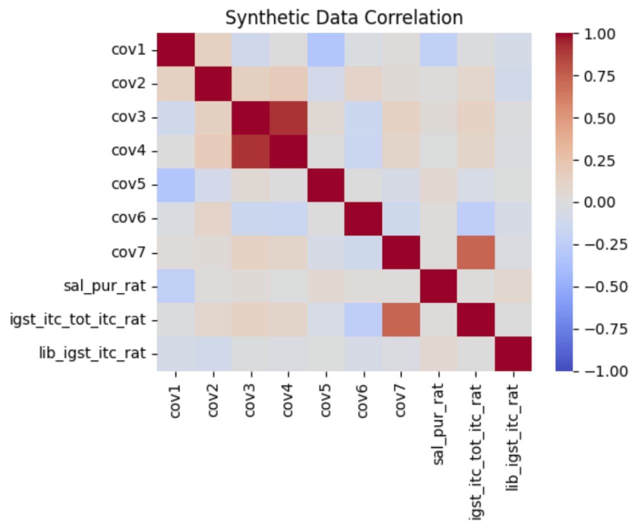
Figure 2: Density comparison plots for each feature: real data vs. synthetic data.

## 4.5 Visualization Outputs

To visually present the correlation analyses for both the real and synthetic datasets, the following figures are separate floats:

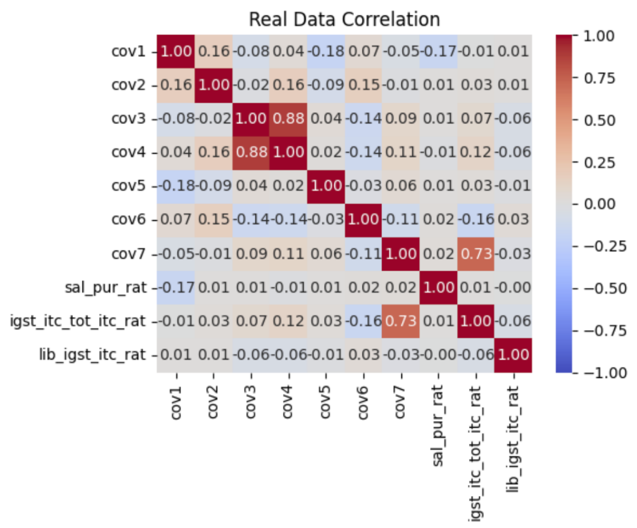


(a) Real Data Correlation

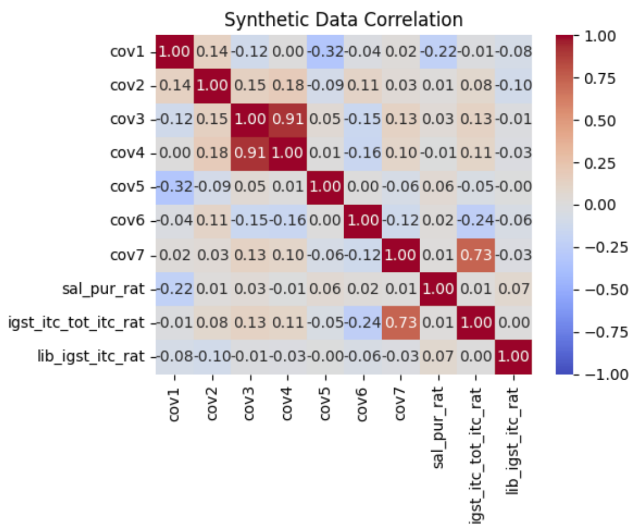


(b) Synthetic Data Correlation

Figure 3: Correlation heatmaps for the real and synthetic datasets.



(a) Real Data Heatmap



(b) Synthetic Data Heatmap

Figure 4: Detailed heatmaps for the real and synthetic datasets.

## 5 Conclusion

This project demonstrates the feasibility of generating synthetic data using a GAN with a Wasserstein loss and gradient penalty. The evaluations via t-SNE, KDE plots, and correlation heatmap analysis suggest that the synthetic data largely captures the structural and statistical properties of the original dataset. The high correlation similarity score further indicates that the synthetic data closely mirrors the relationships among features present in the real data. Such synthetic data can be useful for tasks where data privacy, augmentation, or simulation is required.