Generation of Synthetic Data with GAN

Fraud Analytics Project Report

Sanskriti Agarwal	**K**ocherla Sai Kiran	**K**ota Dhana Lakshmi
CS24MTECH14002	**AI**24MTECH02003	**AI**22BTECH11012

Instructor: Prof. Sobhan Babu

Institution: IIT Hyderabad

April 16, 2025

Contents

Abstract

In many real-world applications, the availability of high-quality, balanced data is a critical bottleneck for developing reliable machine learning models. Synthetic data generation offers a promising solution by enabling the augmentation of datasets where real data is scarce, sensitive, or imbalanced. In this work, we present a robust pipeline for synthetic tabular data generation that integrates outlier detection, data normalization, and advanced generative modeling using Wasserstein GAN with Gradient Penalty (WGAN-GP). Initially, Z-score-based filtering is applied to remove outliers and ensure cleaner input distributions. The preprocessed data is then used to train a WGAN-GP model, which benefits from enhanced training stability and better convergence. The generator and critic networks are designed using fully connected and LeakyReLU layers, respectively. We further incorporate feature matching and correlation matching losses into the generator's objective to preserve important statistical relationships within the data. This ensures the synthetic samples not only mimic the real data distribution but also maintain underlying data structures. Our methodology yields high-quality synthetic data, showing strong alignment with real data in terms of feature and correlation statistics. The proposed pipeline is generalizable, scalable, and applicable across various domains where data availability or privacy is a concern.

1 Problem Statement

In this work, we address the challenge of generating high-fidelity synthetic tabular data using a GAN framework and rigorously evaluating its statistical fidelity against the original dataset. Specifically, we train a GAN on our real-world dataset and then assess the quality of the generated samples through two complementary analyses:

1. We overlay the marginal distributions of each feature for both real and synthetic data to visually inspect whether the GAN has accurately captured individual feature statistics (e.g., means, variances, skewness, and multimodality).

2. We compute and visualize the pairwise Pearson correlation matrices for the real and synthetic datasets—rendered as side-by-side heatmaps—to determine if the synthetic data preserve the inter-feature dependencies present in the original data.

By combining distributional and correlation-based evaluations, our approach provides a comprehensive measure of how well the GAN replicates both the univariate and multivariate structure of the source data, which is critical for downstream tasks that rely on realistic data representations.

2 Dataset Description

The dataset used in this study consists of 1,199 records with 10 continuous numerical features. Each record represents a data sample with the following attributes:

- **cov1** to **cov7**: A set of seven covariates representing various numerical indicators or features.
- **sal_pur_rat**: A financial ratio derived from sales and purchases, likely indicating economic activity or profitability.
- **igst_itc_tot_itc_rat**: A ratio involving IGST Input Tax Credit relative to total Input Tax Credit, which reflects tax-related accounting metrics.
- **lib_igst_itc_rat**: Another tax-related ratio involving liabilities and IGST Input Tax Credit.

All features are of type `float64`, and there are no missing values in the dataset. This makes it well-suited for training GAN models without requiring imputation or preprocessing to handle null values. The continuous nature of the features allows for direct application of GAN-based techniques for synthetic data generation and statistical comparison.

Table 1: First 5 rows of the dataset

cov1	cov2	cov3	cov4	cov5	cov6	cov7	sal_pur_rat	igst_itc_tot_itc_rat	lib_igst_itc_rat
0.9978	0.9999	0.2159	0.1967	0.0000	0.9556	0.9988	-0.0326	1.7618	-0.0543
0.9940	0.9799	-0.3371	-0.2486	0.0000	0.6408	0.5539	-0.0320	-0.6293	-0.0535
0.9476	0.4557	0.0017	0.1286	-0.0041	-0.1621	0.9606	-0.0302	1.5357	-0.0542
0.3966	0.9199	0.4965	0.5768	-0.3407	0.8024	0.6737	-0.0321	0.4492	-0.0541
0.9999	0.3276	0.7005	0.3156	0.0000	0.3008	0.9790	-0.0322	1.7620	-0.0543

3 Methodology

We use a Generative Adversarial Network (GAN) to generate high-quality synthetic tabular data that mirrors the statistical structure of a real-world dataset. Specifically, we adopt a Wasserstein GAN with Gradient Penalty (WGAN-GP) for stable and efficient training. The data is preprocessed by removing outliers via z-score filtering and standardizing features using `StandardScaler`. The GAN consists of a generator that maps random noise to synthetic samples and a critic that scores data realism.

To improve data fidelity, the generator's loss combines three components: adversarial loss, **feature matching loss** to align intermediate critic features, and **correlation matching loss** to preserve inter-feature dependencies. After training, we assess synthetic data quality through distribution plots and Pearson correlation heatmaps. The final synthetic data is inverse-transformed and saved for further use.

Network Architecture

- **Generator** $G(z)$: A fully connected neural network that maps a latent noise vector $z \sim \mathcal{N}(0, I)$ of dimension 32 to a synthetic data point with the same feature dimension as the real dataset.
- **Critic** $D(x)$: A neural network that replaces the traditional discriminator. It outputs a real-valued scalar score representing how "real" the input data appears. Additionally, it outputs intermediate layer activations used for feature matching.

Loss Functions

- **Wasserstein Critic Loss**:

$$\mathcal{L}_D = -\mathbb{E}[D(x_{\text{real}})] + \mathbb{E}[D(x_{\text{fake}})] + \lambda_{\text{GP}} \cdot \text{GP}$$

  where the gradient penalty term GP is defined as:

$$\text{GP} = \mathbb{E}\left[(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1)^2\right]$$

  and $\hat{x}$ is a point interpolated between real and fake samples.

- **Generator Loss**:

$$\mathcal{L}_G = -\mathbb{E}[D(x_{\text{fake}})] + \lambda_{\text{FM}} \cdot \mathcal{L}_{\text{FM}} + \lambda_{\text{Corr}} \cdot \mathcal{L}_{\text{Corr}}$$

  where:

  - **Feature Matching Loss**:

$$\mathcal{L}_{\text{FM}} = \|\mu(D_f(x_{\text{real}})) - \mu(D_f(x_{\text{fake}}))\|_2^2$$

    matches the mean of the intermediate critic features.
  - **Correlation Matching Loss**:

$$\mathcal{L}_{\text{Corr}} = \|\text{Corr}(x_{\text{real}}) - \text{Corr}(x_{\text{fake}})\|_F^2$$

    matches the Pearson correlation structure between features of the real and synthetic data.

4 Optimization Approaches

The GAN architecture employed several optimization strategies to ensure training stability and improve the quality of the synthetic data:

- **Gradient Penalty (WGAN-GP):** A gradient penalty term was included in the discriminator loss to enforce the 1-Lipschitz constraint. This improves convergence stability and prevents mode collapse.
- **Standardization of Input Features:** All features were standardized using z-score normalization. This ensures uniform scale across features, aiding faster and more balanced training.
- **Outlier Filtering:** Outliers were filtered using z-score thresholding before training. Removing extreme values prevents skewing the loss function and helps the generator focus on the representative data distribution.
- **Stable Training Schedule:** The generator and discriminator were updated in alternating steps with separate optimizers (Adam) and tuned hyperparameters ($\beta_1 = 0.5$, $\beta_2 = 0.9$), following best practices for GAN training.
- **Batch Normalization in Generator:** Batch normalization layers were used in the generator to stabilize training and reduce internal covariate shift.

These approaches collectively contributed to producing synthetic data that aligns well with the original distribution and preserves statistical relationships.

5 Tools and Libraries Used

The implementation of the synthetic data generation and evaluation pipeline is done using the Python programming language along with the following key libraries:

- **PyTorch** (`torch`): Used to implement and train the GAN model, including the generator, critic, and training loop with custom loss functions.
- **NumPy** (`numpy`): For numerical operations such as array manipulations, statistics, and sampling from noise distributions.
- **Pandas** (`pandas`): For loading, cleaning, and handling tabular data, including DataFrame operations and exporting CSV files.
- **scikit-learn** (`sklearn.preprocessing.StandardScaler`): Used to standardize the dataset features to zero mean and unit variance before feeding them into the GAN.
- **Matplotlib** (`matplotlib.pyplot`): For visualizing the generator and critic loss curves across epochs.
- **Seaborn** (`seaborn`): For plotting feature distributions and correlation heatmaps to compare real and synthetic data.
- **SciPy** (`scipy.stats.zscore`): For identifying and removing outliers using the z-score method.

6 High-Quality Pseudocode

Algorithm 1 Preprocessing Step

procedure PREPROCESSING($\mathcal{D}$)
    for each column c in $\mathcal{D}$ with known outliers **do**
        Remove rows where $|z\text{-score}(c)| > \text{threshold}_c$
    end for
    Normalize $\mathcal{D}$ using standard scaling
end procedure

Algorithm 2 Model Initialization

procedure INITIALIZEMODELS
    Initialize Generator $G(z)$ with fully-connected layers
    Initialize Critic $D(x)$ with LeakyReLU layers
    Set hyperparameters: $\lambda_{gp}, \lambda_{fm}, \lambda_{corr}, \text{lr}, n_{critic}, n_{epochs}$
end procedure

Algorithm 3 Training WGAN-GP

procedure TRAINWGAN-GP
    for epoch $= 1$ to n_{epochs} **do**
        for each batch x in DataLoader **do**
            for $i = 1$ to n_{critic} **do**
                Sample noise $z \sim \mathcal{N}(0, I)$
                $\tilde{x} \leftarrow G(z)$
                Compute $D(x)$ and $D(\tilde{x})$
                $GP \leftarrow \text{Penalty}(D, x, \tilde{x})$
                $L_D \leftarrow -(\mathbb{E}[D(x)] - \mathbb{E}[D(\tilde{x})]) + \lambda_{gp} \cdot GP$
                Update D
            end for
            Sample $z \sim \mathcal{N}(0, I)$
            $\tilde{x} \leftarrow G(z)$
            Compute features $f_{\text{real}}, f_{\text{fake}}$ using D
            Compute correlations $\rho_{\text{real}}, \rho_{\text{fake}}$
            $L_G \leftarrow -\mathbb{E}[D(\tilde{x})] + \lambda_{fm} \cdot \text{MSE}(f_{\text{real}}, f_{\text{fake}}) + \lambda_{corr} \cdot \text{MSE}(\rho_{\text{real}}, \rho_{\text{fake}})$
            Update G
        end for
    end for
end procedure

Algorithm 4 Generating Synthetic Data

procedure GENERATESYNTHETICDATA
    Sample $z \sim \mathcal{N}(0, I)$
    $\hat{\mathcal{D}} \leftarrow G(z)$
    return $\hat{\mathcal{D}}$
end procedure

7 Results and Visualization Interpretation

7.1 Visualization Explanation

Figure 1 presents the distribution comparison between real and synthetic data for selected features. The close alignment of the distributions indicates that the GAN successfully captured the marginal feature behavior without overfitting.

Figure 2 shows the Pearson correlation heatmaps of the real and synthetic datasets. The synthetic data preserves the key correlation structures present in the original data, suggesting the generator effectively learned the inter-feature relationships with stability.

7.2 Visualization Plots

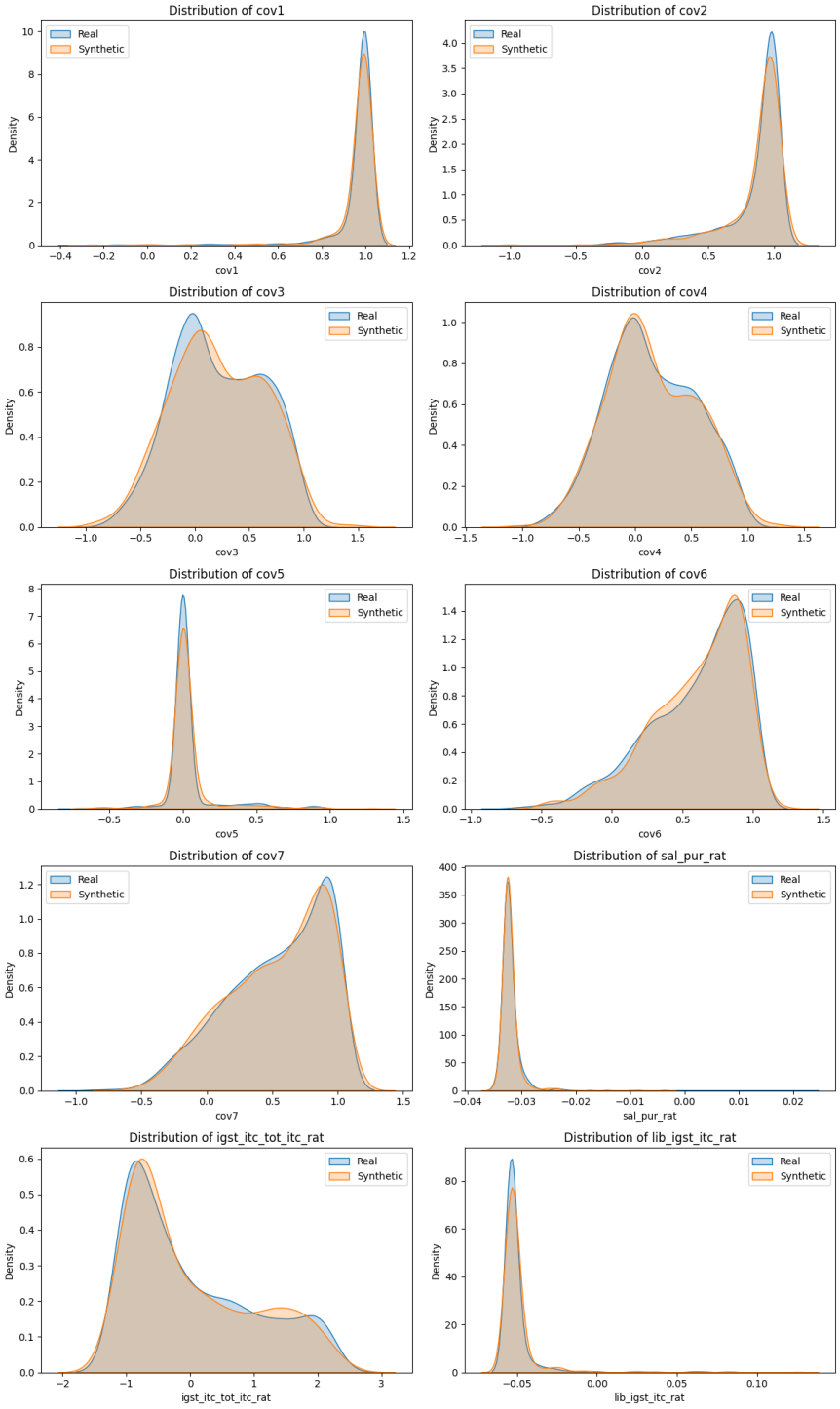

Figure 1: Distribution of all features for both the real and synthetic datasets

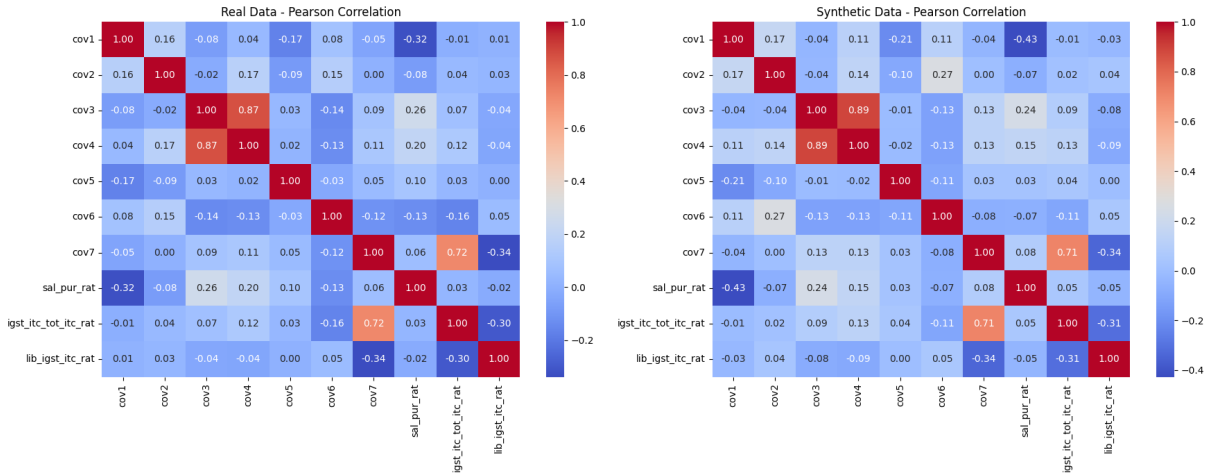

Figure 2: Comparison of correlation matrices for both the real and synthetic datasets

8 Conclusion

In this work, we presented a comprehensive pipeline for synthetic data generation using a Wasserstein GAN with Gradient Penalty (WGAN-GP), tailored for structured tabular data. We began by enhancing data quality through statistical outlier removal using Z-score filtering, which ensured that extreme values did not distort the learned distribution. The WGAN-GP framework offered stable and effective adversarial training by enforcing the Lipschitz constraint via gradient penalty. To further improve generation fidelity, we incorporated feature matching loss—ensuring similarity in intermediate feature representations between real and synthetic data—and correlation matching loss to preserve inter-feature dependencies via Pearson correlation matrices. Together, these components enabled the generation of high-quality synthetic samples that are statistically consistent and structurally similar to the original data. This method is particularly useful in domains like fraud detection, where data privacy and augmentation are crucial.

References

[1] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, *Improved Training of Wasserstein GANs*, in Advances in Neural Information Processing Systems (NeurIPS), 2017.

[2] E. Choi, S. Biswal, B. Malin, J. Duke, W. Stewart, and J. Sun, *Generating Multi-label Discrete Patient Records using Generative Adversarial Networks*, in Machine Learning for Healthcare Conference (MLHC), 2017.

[3] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, *Modeling Tabular Data using Conditional GAN*, in Advances in Neural Information Processing Systems (NeurIPS), 2019.

[4] S. Zhao, Z. Song, and S. Ermon, *Inferring Rewards from Language with Context-Aware Inverse Reinforcement Learning*, in International Conference on Learning Representations (ICLR), 2019.

[5] C. C. Aggarwal, *Outlier Analysis*, Springer, 2nd ed., 2017.