People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

KASDI MERBAH UNIVERSITY - OUARGLA

Faculty of New Technologies of Information and Communication

Department of Computer Science and Information Technology

MASTER THESIS

Domain: Mathematics and Computer Science
Field: Computer Science
Speciality: **Artificial Intelligence and Data Science**

Theme

Fuzzy Logic-Based WGAN-GP Data Augmentation and Hybrid DL-ML Models for Robust Credit Card Fraud Detection

By: Ghiaba Souhila and Benamar Ritadj

Jury Members:

Supervisor: Said Bachir

President: Bouanane Khadra

Examiner: Benchabana Ayoub

Academic Year: 2024/2025

Acknowledgments

First and foremost, we praise and thank Allah Almighty for His guidance and facilitation throughout our academic journey, and for His countless blessings and grace that supported and helped us overcome all difficulties and challenges.

We would also like to extend our sincere gratitude and deep appreciation to our esteemed supervisor, Dr. Said Bachir, for his continuous support, valuable guidance, and constant encouragement throughout the preparation of this thesis. His precise instructions and commitment to excellence played a crucial role in achieving this honorable academic outcome.

We cannot forget to express our deep thanks to all the professors and teachers who had a significant impact on our academic path. Their dedication to their work and love for their profession were a source of inspiration and motivation for us to continue striving for success.

We also owe our thanks to our dear families, who have always been a pillar of support with their love, patience, and sacrifices. Their prayers and continuous encouragement were the fuel that kept us going to achieve our goals.

Finally, we thank our dear friends for their moral support, understanding, and for sharing this journey with us. Their companionship was a great help in overcoming challenges and a constant source of joy and positive energy.

Abstract

Credit card fraud is one of the most serious threats facing modern financial systems, causing losses estimated in billions of dollars annually worldwide. Traditional fraud detection systems face significant challenges, one of the most critical is the class imbalance problem, where fraudulent transactions account for less than 0.1% of total transactions, making learning and classification processes difficult.

This study aims to develop a hybrid system based on deep learning techniques to address this issue by adopting an advanced data generation approach using Wersstein Generative Adversarial Networks Gradient penalty (WGAN-GP), supported by Spectral Normalization and Fuzzy Logic. The proposed system generates realistic synthetic fraudulent transactions, to augment the dataset, addressing class imbalance and improving minority class representation.

Several deep learning models were trained on the augmented data, including Convolutional Neural Networks (CNN), Long Short-Term Memory Networks (LSTM), and the XGBoost algorithm. The performance was evaluated using standard metrics such as the F1-score and the Fréchet Inception Distance (FID) to assess the quality of the generated data.

Experimental results demonstrated that integrating WGAN-GP with fuzzy logic significantly improves the accuracy and efficiency of fraud detection, proving the effectiveness of the proposed model in handling data imbalance and uncertainty in real-world scenarios.

Keywords: Credit Card Fraud Detection, Class Imbalance, Deep Learning, WGAN-GP, Spectral Normalization, Fuzzy Logic, Data Augmentation, CNN, LSTM, XGBoost.

Résumé

La fraude à la carte bancaire est l'une des menaces les plus graves auxquelles sont confrontés les systèmes financiers modernes, entraînant des pertes estimées à des milliards de dollars chaque année dans le monde. Les systèmes traditionnels de détection de la fraude rencontrent des défis importants. L'un des plus critiques est le problème du déséquilibre des classes, où les transactions frauduleuses représentent moins de 0,1% du total, ce qui rend les processus d'apprentissage et de classification difficiles.

Cette étude vise à développer un système hybride basé sur des techniques d'apprentissage profond pour répondre à ce problème en adoptant une approche avancée de génération de données à l'aide des réseaux antagonistes génératifs avec pénalité de gradient (WGAN-GP), renforcée par la normalisation spectrale et la logique floue. Le système proposé génère des transactions frauduleuses synthétiques réalistes afin d'augmenter le jeu de données, de corriger le déséquilibre des classes et d'améliorer la représentation de la classe minoritaire.

Plusieurs modèles d'apprentissage profond ont été entraînés sur les données augmentées, notamment les réseaux de neurones convolutifs (CNN), les réseaux de mémoire à long terme (LSTM) et l'algorithme XGBoost. Les performances ont été évaluées à l'aide de métriques standards telles que le F1-score et la distance Fréchet Inception (FID) pour évaluer la qualité des données générées.

Les résultats expérimentaux ont démontré que l'intégration de WGAN-GP avec la logique floue améliore de manière significative la précision et l'efficacité de la détection de fraude, prouvant l'efficacité du modèle proposé face au déséquilibre des données et à l'incertitude dans des scénarios réels.

Mots-clés : Détection de fraude à la carte bancaire, Déséquilibre des classes, Apprentissage profond, WGAN-GP, Normalisation spectrale, Logique floue, Augmentation de données, CNN, LSTM, XGBoost.

ملخص

تُعدّ عمليات الاحتيال باستخدام بطاقات الائتمان من أخطر التهديدات التي تواجه الأنظمة المالية الحديثة، حيث تتسبب في خسائر تُقدّر بمليارات الدولارات سنويًا على مستوى العالم. تواجه أنظمة كشف الاحتيال التقليدية تحديات كبيرة، من أبرزها مشكلة اختلال التوازن بين الفئات، إذ لا تمثل المعاملات الاحتيالية سوى أقل من 0,1% من إجمالي المعاملات، مما يجعل عمليات التعلم والتصنيف صعبة للغاية.

تهدف هذه الدراسة إلى تطوير نظام هجين يعتمد على تقنيات التعلم العميق لمعالجة هذه الإشكالية، من خلال اعتماد نهج متقدم لتوليد البيانات باستخدام شبكات الخصومة التوليدية من نوع **WGAN-GP** المدعّمة بتقنيات **Spectral Normalization** و **Fuzzy Logic**. يعمل النظام المقترح على توليد معاملات احتيالية اصطناعية واقعية تُستخدم لاحقًا لموازنة البيانات وتحسين تمثيل الفئة النادرة.

تم تدريب عدة نماذج تعلم عميق على البيانات المُعزَّزة، بما في ذلك **الشبكات العصبية الالتفافية (CNN)**، و **شبكات الذاكرة طويلة المدى (LSTM)**، إلى جانب خوارزمية **XGBoost**. وقد تم تقييم الأداء باستخدام معايير قياسية مثل **معدل F1** و **مسافة فريشيت للتقاطع (FID)** لقياس جودة البيانات المُولَّدة.

أظهرت النتائج التجريبية أن دمج **WGAN-GP** مع تقنيات المنطق الضبابي يساهم بشكل كبير في تحسين دقة وكفاءة كشف الاحتيال، مما يؤكد فعالية النموذج المقترح في التعامل مع مشكلتي اختلال التوازن وعدم اليقين في بيئات العالم الحقيقي.

الكلمات المفتاحية: كشف الاحتيال باستخدام بطاقات الائتمان، اختلال التوازن بين الفئات، التعلم العميق، WGAN-GP، التطبيع الطيفي، المنطق الضبابي، تعزيز البيانات، الشبكات العصبية الالتفافية(CNN) ، شبكات(LSTM) الذاكرة طويلة المدى ،XGBoost.

Contents

List of Tables

List of Figures

1.General Introduction

1.1 Introduction

The severity of financial fraud is considered one of the fundamental issues threatening the financial security of both individuals and corporations. In addition to the substantial financial losses incurred as a result of such activities, financial fraud also contributes to the erosion of trust in digital financial systems. This, in turn, may negatively impact the ability of financial institutions to attract clients and preserve their reputations.

Among the most prevalent forms of financial fraud is credit card fraud, which is responsible for billions of dollars in losses annually. For example, losses resulting from this type of card payment fraud amounted to approximately \$33.8 billion in 2023 alone. According to projections by the Nilson Report, these losses are expected to increase significantly, potentially reaching \$403.88 billion over the next decade—clearly reflecting the growing severity of this global threat.[28].

Despite notable advancements in fraud detection technologies, particularly those based on machine learning and deep learning, numerous challenges persist for researchers and practitioners in the field. One of the most critical challenges is the data imbalance problem, which complicates model training and evaluation. Additionally, the continuous evolution of fraudsters' techniques poses significant difficulties for traditional detection systems, limiting their ability to adapt swiftly and effectively.

1.2 Problem Statement And Motivation

Credit card fraud detection systems face significant technical challenges, the most critical of which is the class imbalance problem, where fraudulent transactions constitute a very small fraction of total transactions—often less than 0.1%. This severe imbalance causes machine learning models to be trained on biased data, leading to a focus on correctly classifying legitimate transactions while failing to detect the rare fraudulent ones. Consequently, the false negative rate increases considerably.

This issue has a direct impact on real-world applications, where many fraudulent transactions remain undetected in time, resulting in substantial financial losses and a decline in user trust in digital payment systems. Although several methods, such as the Synthetic Minority Over-sampling Technique (SMOTE) and loss function adjustments, have been proposed to mitigate class imbalance, their performance often remains limited, especially in dynamic environments where fraud strategies continuously evolve.

Therefore, this study aims to address the class imbalance problem by developing a robust and intelligent model using advanced learning techniques capable of handling imbalanced data and improving the detection accuracy of fraudulent transactions.

1.3 objectives

The main objective of this study is to address the problem of data imbalance in credit card fraud detection by developing an advanced data augmentation framework based on Generative Adversarial Networks (GANs).By integrating Spectral Normalization with a GAN architecture utilizing Wersstein Gradient Penalty (WGAN-GP).The generated data is then used to train various machine learning models, and the performance is evaluated using F1-score and Fréchet Inception Distance (FID) to ensure the effectiveness and realism of the synthetic data.

1.4 Contributions

This study presents a comprehensive approach to improve credit card fraud detection in the context of highly imbalanced datasets. The main contributions of this work are as follows:

1. An advanced data augmentation framework was developed using **Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP)**, enhanced by **Spectral Normalization** and **Fuzzy Logic**, to generate realistic synthetic samples of the minority class (fraudulent transactions).

2. **Fuzzy Logic** was integrated into the generative process to introduce uncertainty modeling and enhance the quality and diversity of the generated fraudulent samples.

3. The quality of the generated data was assessed using the **Fréchet Inception Distance (FID)** metric to ensure statistical similarity between real and synthetic data samples.

4. **Deep learning models**, specifically **Convolutional Neural Networks (CNN)** and **Long Short-Term Memory networks (LSTM)**, were employed for feature extraction from transaction data.

5. Extracted features were classified using the **XGBoost** algorithm, creating a hybrid architecture that combines the representation power of deep learning with the classification strength of machine learning.

6. Extensive experiments were conducted using standard evaluation metrics such as precision, recall, F1-score, and confusion matrix across multiple datasets to validate the effectiveness and robustness of the proposed framework.

7. Comparative analysis with state-of-the-art methods demonstrated that the proposed approach significantly improves fraud detection performance, particularly in detecting minority class instances under severe class imbalance.

1.5 Thesis Overview

Chapter 1: Introduction

This chapter provides an overview of the research, highlighting its motivations, objectives, and main contributions to its field. It also presents a brief background on the adopted methodologies.

Chapter 2: Related Work

This chapter lays the foundation for the research by reviewing previous studies related to credit card fraud detection, with a focus on the methods and models used in this field.

Chapter 3: Materials and Methodologies

This chapter provides a comprehensive review of the materials and methodologies employed in this research to address the issue of credit card fraud detection, detailing the tools and techniques utilized throughout the study.

Chapter 4: Results

This chapter presents the results obtained through the implementation of the proposed solutions to address the credit card fraud detection problem, highlighting their effectiveness.

Chapter 5: Conclusions

This chapter presents the conclusions drawn from the research and experiments conducted.

1.6 Background

In this part, we highlight the methods employed to address the issue of data imbalance, as well as the algorithms used to detect credit card fraud. General definitions of each method are also provided to offer a clear understanding of the techniques adopted in this study

1.6.1 Credit Card Fraud Detection - CCFD

Is a field within data science and artificial intelligence concerned with analyzing credit card transaction data to quickly and accurately identify fraudulent or illegal activities. CCFD techniques rely on machine learning algorithms such as Neural Networks and Long Short-Term Memory Networks (LSTM), in addition to other approaches like Semi-Supervised Learning and Anomaly Detection, to identify transactions that deviate from the cardholder's normal behavior.

The fraud detection process depends on diverse data types, including: Transaction ID, Timestamp, Amount, Location, Transaction Type (whether online, in-store, or ATM withdrawal), as well as additional features like Merchant Category, Device ID, and the cardholder's spending patterns.

The concept behind CCFD is based on the observation that fraudulent activities often exhibit patterns different from normal activities. For instance, a sudden large purchase in a foreign country might be a strong indicator of fraud. Therefore, the main goal is to build an intelligent model capable of distinguishing between legitimate and fraudulent transactions based on transaction features, while ensuring decisions are made without delay that could negatively impact the user experience.

Despite advancements, CCFD faces several major challenges. One of the most prominent is the Imbalanced Data problem, where fraudulent transactions represent only a very small fraction of all transactions, making model training more complex as algorithms may overlook rare fraudulent cases. Additionally, Concept Drift poses another challenge, as fraud patterns evolve over time, requiring continuous model updates to maintain effectiveness.

Moreover, fraud detection demands Real-Time Decision-Making, necessitating the development of fast-response systems capable of issuing decisions within seconds of transaction execution. Another obstacle is the Scarcity of Labeled Data, where not all transactions are accurately labeled as fraudulent or legitimate, making it necessary to employ semi-supervised or unsupervised learning techniques. Finally, Privacy and Confidentiality must be maintained, imposing additional constraints on how sensitive customer data is collected, processed, and stored.

1.6.2 Machine learning Architectures

In credit card fraud detection (CCFD), various machine learning (ML) techniques are employed to identify fraudulent transactions with high precision and recall. Among these, **Extreme Gradient Boosting (XGBoost)** is widely recognized for its performance and scalability. However, it is important to note that XGBoost is not the only ML technique used in CCFD. Other models, including **Logistic Regression**, **Random Forests**, and **Support Vector Machines (SVM)**, are also commonly applied depending on the nature of the dataset and specific detection goals.

Extreme Gradient Boosting -XGBoost

Is a machine learning algorithm based on the Gradient Boosting technique, developed to improve speed and accuracy compared to traditional techniques. It was first introduced by Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). [8]. The algorithm works by building a sequence of decision tree models in a stepwise manner, where each new model attempts to correct the errors of the previous models by optimizing the loss function using gradient descent. XGBoost features several enhancements, including advanced regularization techniques (L1 and L2) to prevent overfitting, parallel processing support to speed up tree construction, and smart handling of missing values, making it highly suitable for large-scale data.

The working mechanism starts with selecting an appropriate loss function based on the nature of the task, followed by calculating the residuals (errors) after each model. A new tree is then created to correct these errors, and the predictions are updated progressively using an equation that adds the output of the new tree to the previous predictions, with a learning rate to control the size of the updates.

The core update equation is as follows:

$$y_{\text{pred}}^{(t)} = y_{\text{pred}}^{(t-1)} + \eta f_t(x) \tag{1.1}$$

where:

- $y_{\text{pred}}^{(t)}$: the prediction at iteration t.
- $y_{\text{pred}}^{(t-1)}$: the prediction at the previous iteration.

- η: the learning rate, which controls the size of the updates.
- $f_t(x)$: the model learned at iteration t.

To enhance model performance and prevent excessive complexity, XGBoost incorporates regularization terms within the objective function. The algorithm also supports early stopping during training if the performance on the validation set doesn't improve after a certain number of iterations, which helps improve model efficiency and reduce resource consumption. Due to these features, XGBoost has become one of the most popular and effective machine learning tools in competitions and real-world applications.

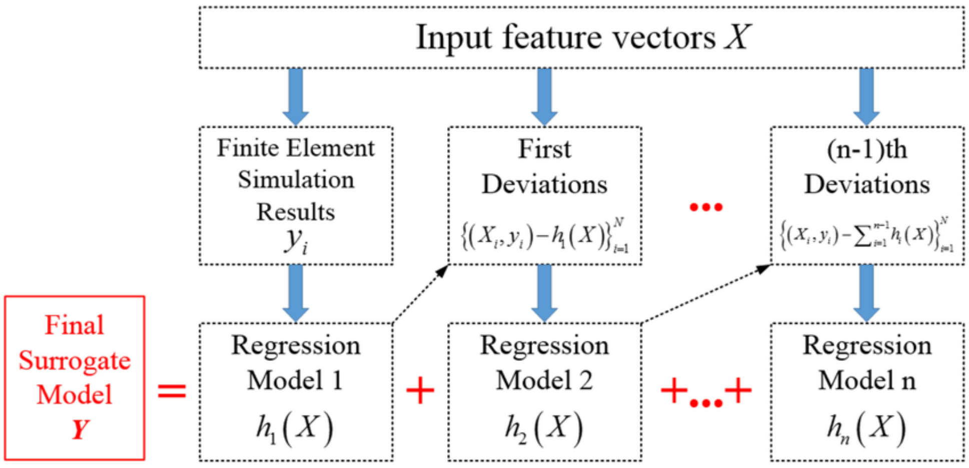

Input feature vectors X
Finite Element Simulation Results y_i
First Deviations $\{(X_i, y_i) - h_1(X)\}_{i=1}^{N}$
(n-1)th Deviations $\{(X_i, y_i) - \sum_{i=1}^{n-1} h_i(X)\}_{i=1}^{N}$
Final Surrogate Model Y
=
Regression Model 1 $h_1(X)$
+
Regression Model 2 $h_2(X)$
+...+
Regression Model n $h_n(X)$

Figure 1.1: Extreme Gradient Boosting - XGBoost Architectures [31]

1.6.3 Deep Learning Architectures

Generative Adversarial Networks - GANs

Generative Adversarial Networks (GANs) are an advanced framework within deep learning used to generate new data that resembles real data. This is achieved by training two neural networks in a competitive manner. GANs were first proposed in 2014 by Ian Goodfellow and his team, and they are considered powerful tools in various domains such as image generation, data augmentation, and replicating complex patterns in data.[14]

A GAN model consists of two main neural networks:

.Generator: The Generator is responsible for producing synthetic data that closely approximates real data. It begins by sampling a random noise vector, which represents a point in the latent space, and transforms this vector through a series of neural network layers into data with statistical properties resembling those of the real dataset.

The generator is trained in a gradual and adversarial manner by leveraging feedback from the Discriminator—the second component of the Generative Adversarial Network (GAN)—which evaluates the authenticity of the generated samples. As training progresses and the quality of the outputs improves, the generator becomes increasingly effective at deceiving the discriminator, making its synthetic data nearly indistinguishable from real data.

Architecturally, the generator is typically composed of either fully connected (dense) layers or convolutional layers, depending on the nature of the task. Activation functions such as ReLU (Rectified Linear Unit) are often used in intermediate layers, while functions like Tanh are employed in the output layer, particularly when the data is normalized.

Through iterative training, the generator learns to capture the underlying distributions and structural patterns of the real data, ultimately enabling the production of realistic and high-fidelity samples.

.Discriminator: A neural network that functions as a binary classifier, with the primary objective of distinguishing between real data samples—sourced from the original dataset—and fake samples generated by the Generator. Its core role is to evaluate the realism of the input samples and provide feedback to the Generator to improve the quality of its outputs.

The Discriminator processes each sample, whether real or generated, by passing it through a sequence of neural network layers, which may be either fully connected (dense) or convolutional, depending on the nature of the input data. Nonlinear activation functions, such as LeakyReLU, are commonly used in the hidden layers due to their ability to mitigate the vanishing gradient problem. The final layer typically employs an activation function that outputs a probability value, indicating the degree of realism of the input—closer to one if the sample is likely real.

During training, the Discriminator learns to accurately differentiate between real and fake samples, and it relays this evaluation to the Generator as feedback to enhance its performance. As the Generator improves and begins to produce more convincing samples, the Discriminator's task becomes increasingly difficult. This leads to a progressive balance between the two models. Eventually, if the Discriminator can no longer reliably distinguish between real and generated data, it is considered a strong indication that the Generator has succeeded in producing highly realistic data.

It is worth noting that the adversarial relationship between the Generator and the Discriminator in a GAN model is mathematically formulated through an objective function, also referred to as the loss function. This function represents a zero-sum game between the two models, where the Discriminator aims to maximize its ability to distinguish between real and fake data, while the Generator seeks to minimize the Discriminator's ability to do so.

The following mathematical expression illustrates this relationship:

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

- G: Generator, creates fake samples.
- D: Discriminator, distinguishes between real and generated data.
- $x \sim p_{data}(x)$: Real data drawn from the true data distribution.
- $z \sim p_z(z)$: Random noise vector fed into the generator.
- $G(z)$: The sample generated by the generator using the noise.
- $D(x)$: Probability that the sample x is real (calculated by the discriminator).
- $V(D,G)$: The objective or loss function that is optimized in the competitive game between G and D.

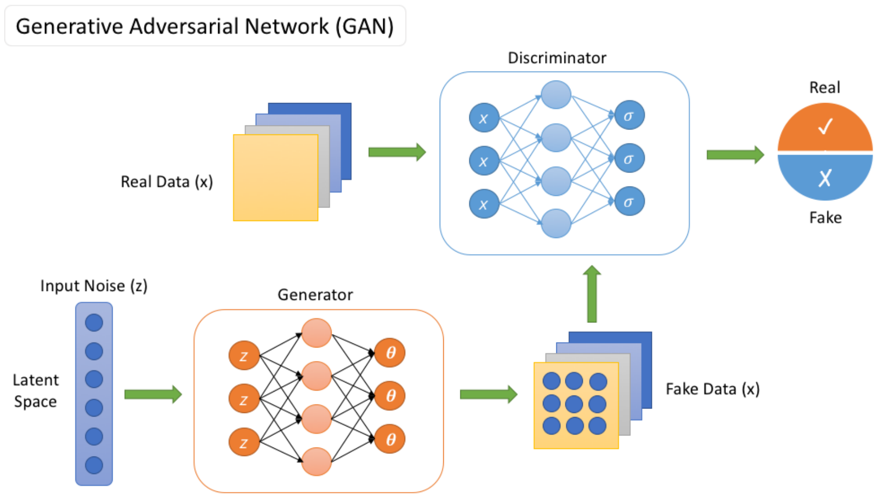


Figure 1.2: GANs Architectures [34]

Wasserstein Generative Adversarial Network-WGAN Wasserstein GAN is a deep generative model that builds upon the traditional Generative Adversarial Network (GAN) framework, but reformulates the loss function using the *Wasserstein-1 distance* (also known as the *Earth Mover's Distance*) instead of conventional divergence measures such as KL divergence or JS divergence. This reformulation aims to address common issues associated with traditional GANs, such as *mode collapse* and *vanishing gradients*. The Wasserstein-1 distance is differentiable and provides meaningful gradients even when there is little to no overlap between the real data distribution $\mathbb{P}_r$ and the generator's distribution $\mathbb{P}_g$. WGAN was first introduced in the seminal paper *"Wasserstein GAN"* by Arjovsky et al. (2017), and it represented a significant advancement in generative modeling by improving training stability and enhancing the quality of generated samples.

WGAN is based on the following theoretical formulation:

$$W(\mathbb{P}_r, \mathbb{P}_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x\sim\mathbb{P}_r}[f(x)] - \mathbb{E}_{x\sim\mathbb{P}_g}[f(x)] \tag{1.2}$$

Where:

- $W(\mathbb{P}_r, \mathbb{P}_g)$ represents the **Wasserstein-1 distance** between the two distributions.
- $\mathbb{P}_r$ denotes the **real data distribution**, i.e., the distribution from which true samples are drawn.
- $\mathbb{P}_g$ denotes the **generated data distribution**, i.e., the distribution obtained by passing random noise through the generator $G(z)$.
- f is a function that satisfies the **1-Lipschitz condition**, meaning its gradient norm is at most 1, which ensures training stability.

Discriminator Loss:

$$\mathcal{L}_D = \mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] - \mathbb{E}_{x\sim\mathbb{P}_r}[D(x)] + \lambda\,\mathbb{E}_{\hat{x}\sim\mathbb{P}_{\hat{x}}}\left(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1\right)^2 \tag{1.3}$$

Generator Loss:

$$\mathcal{L}_G = -\mathbb{E}_{\tilde{x}\sim\mathbb{P}_g}[D(\tilde{x})] \tag{1.4}$$

- $\tilde{x}$: Fake samples from the generator $(\tilde{x} \sim \mathbb{P}_g)$

- x : Real samples from the dataset ($x \sim \mathbb{P}_r$)
- $\hat{x}$: Samples interpolated between real and fake (used for gradient penalty)
- $D(\cdot)$: Output of the discriminator (critic)
- λ : Gradient penalty coefficient

In practice, the function f is approximated by a neural network called the **critic**, instead of a discriminator. Unlike a discriminator, the critic does not classify samples as real or fake, but assigns real-valued scores that reflect how close the generated data is to the real distribution.

Spectral Normalization

Spectral Normalization is an effective regularization technique used in training deep neural networks, particularly applied to the Discriminator in Generative Adversarial Networks (GANs). This technique aims to enhance model stability during training by controlling the amount of gradient amplification caused by weight matrices, which helps to prevent common issues such as exploding or vanishing gradients.

The core idea behind Spectral Normalization is to regulate the spectral norm of the weight matrix in each layer. The spectral norm corresponds to the largest singular value of the weight matrix. This norm is used to normalize the weights, reducing signal fluctuation across layers and enabling a more stable and balanced learning process. Mathematical Formulation: Given a weight matrix $W \in \mathbb{R}^{m \times n}$, the spectral norm is computed as:

$$\|W\|_\sigma = \sup_{\|h\|_2=1} \|Wh\|_2$$

That is, $\|W\|_\sigma$ is the largest singular value of the matrix W. Once the spectral norm is calculated, the weight matrix is normalized as follows:

$$\bar{W} = \frac{W}{\|W\|_\sigma}$$

- **W**: The weight matrix.
- h: A vector (typically of dimension n) that is a unit vector, meaning its Euclidean norm $\|h\|_2$ equals 1.
- Wh: The result of the linear multiplication of the weight matrix W with the vector h, which produces a new vector (typically of dimension m).
- σ refers to the singular values of the matrix W.
- $\|W\|_\sigma$: The spectral norm of the weight matrix, which represents the largest singular value of the matrix.
- sup: The supremum, which refers to the maximum value of $\|Wh\|_2$, i.e., the largest possible length that can result from multiplying W by the unit vector h.
- $\bar{W}$: The weight matrix after being normalized using the spectral norm.

Gradient Penalty

Gradient Penalty is a technique used in deep learning models—specifically in WGAN with Gradient Penalty (WGAN-GP)—to enforce the 1-Lipschitz constraint on the critic function in a smooth and more flexible manner than weight clipping.

This penalty aims to regularize the gradient norm of the critic with respect to its input, encouraging it to remain close to 1. This is achieved by adding a term to the loss function that penalizes deviations from this condition.

The mathematical formulation of the gradient penalty is:

$$\mathcal{L}_{GP} = \lambda \cdot \mathbb{E}_{\hat{x} \sim \mathbb{P}_{\hat{x}}} \left[\left(\|\nabla_{\hat{x}} f(\hat{x})\|_2 - 1 \right)^2 \right]$$

where:

- λ: A regularization coefficient that controls the strength of the penalty (typically set to 10).
- $\hat{x}$: A point interpolated between a real sample and a generated sample.
- f: The critic function.
- $\nabla_{\hat{x}} f(\hat{x})$: The gradient of the critic function with respect to the input $\hat{x}$.

This technique helps ensure that gradients neither vanish (becoming too small) nor explode (becoming too large), which in turn enhances training stability and improves the quality of the generated samples.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are a prevalent and extensively utilized deep learning framework. Initially made for handling visual data, they now work in multiple areas, including structured and sequential data like credit card transactions [17]. The essential structure of a CNN is built from many specialized layers, each serving a specific role in feature extraction and learning. Central to this are the convolutional layers, employing learnable filters (or kernels) that move across the input data. This process captures spatial feature hierarchies by recognizing important local patterns—like edges, shapes, or, in non-image data, relationships among input features. Convolutional actions are succeeded by non-linear activation functions, often ReLU (Rectified Linear Unit). These introduce non-linearity to the model, enabling it to learn complex data representations. [33]

To lessen the data's spatial dimensions and computational burden while maintaining crucial details, pooling layers such as Max Pooling are included. These layers downsample the feature maps and make the model more resistant to slight changes in the input. As the network deepens with more convolutional and pooling layers, it captures increasingly abstract and higher-level features. After this hierarchical feature extraction, multi-dimensional feature maps become a one-dimensional vector and go through fully connected (dense) layers. These layers learn complex feature relationships, generating meaningful predictions.

In the final phase, the output layer generally includes one neuron with a sigmoid activation function for binary classification tasks, like determining if a transaction is fraudulent. The sigmoid function produces a probability between 0 and 1, representing fraud likelihood. What gives CNNs an edge in fraud detection is their ability to automatically learn important features from raw or reshaped data, reducing the need for feature engineering. Their capacity to capture spatial/local dependencies and model non-linear interactions makes them efficient at identifying subtle and complex fraud patterns that traditional models could miss. When transaction data is reshaped into a suitable format for CNN input, such as 2D matrices, these networks can be used to deliver high accuracy, strong performance, and useful detection methods in financial fraud detection systems.

$$a_{i,j}^{(l)} = \sigma\left(\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} w_{m,n}^{(l)} \cdot x_{i+m,j+n}^{(l-1)} + b^{(l)}\right) \tag{1.5}$$

- $a_{i,j}^{(l)}$: Activation output at position (i, j) in layer l
- σ: Activation function (e.g., ReLU, Sigmoid)
- $w_{m,n}^{(l)}$: Weight of the convolution filter at position (m, n) in layer l
- $x_{i+m,j+n}^{(l-1)}$: Input value from the previous layer $(l - 1)$ at position shifted by (m, n)
- $b^{(l)}$: Bias term for layer l
- $M \times N$: Dimensions of the convolution filter (kernel size)

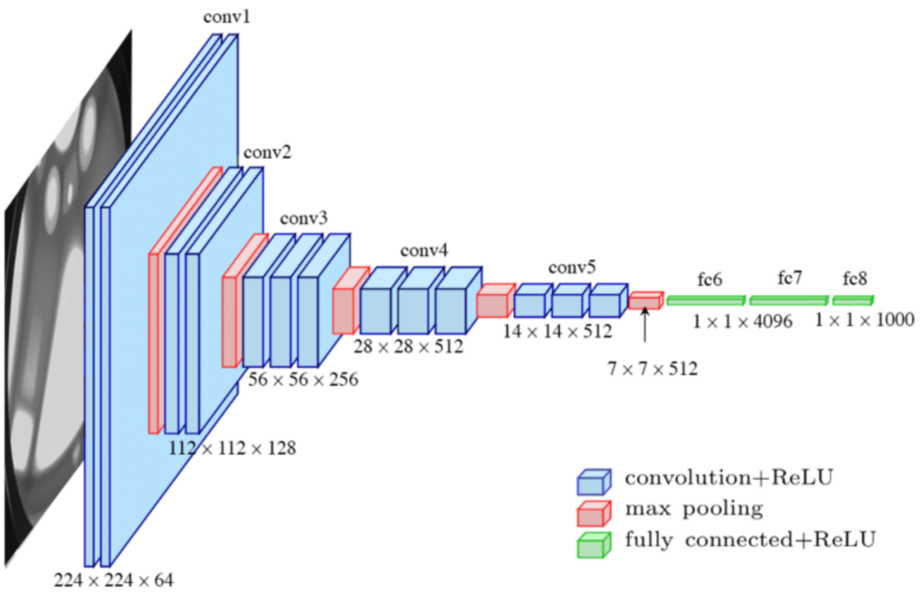


Figure 1.3: CNNs Architectures [9]

Long Short-Term Memory Networks

Long Short-Term Memory networks (LSTMs) represent a type of recurrent neural network (RNN) [10], engineered to surpass the issues that traditional RNNs face when dealing with long-range dependencies within sequential data. Classic RNNs often struggle with vanishing or exploding gradients, hindering their capacity to retain data across numerous time steps. LSTMs address this limitation through a more elaborate design featuring memory cells and three key gates: the input gate, the forget gate, and the output gate. These gates control the information flow throughout the network, giving it the power to selectively store or disregard data over time. The memory cell functions like a data pipeline, maintaining vital information as it processes each time step. The gates then determine which data should be updated, retained, or cleared.

This capability makes LSTMs exceptionally fitting for use with time-series data, sequential patterns, or areas where the past significantly shapes the present – such as speech recognition, language modeling, and financial fraud detection [4]. In the realm of credit card fraud, LSTMs can analyze a customer's transaction history. This allows the network to grasp behavioral patterns and identify discrepancies potentially indicative of fraudulent activities. Consider a customer who usually makes small, local purchases, but unexpectedly initiates a large international transaction. The LSTM can identify this as unusual by analyzing the temporal context. Furthermore, LSTMs can be integrated with dense layers at the output for binary classification—determining whether a transaction is fraudulent. They excel at capturing both short-term and long-term relationships, making them effective at detecting minor alterations in user behavior that other models might fail to notice.

$$\begin{aligned}
f_t &= \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) && \text{(Forget gate)}\\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) && \text{(Input gate)}\\
\tilde{C}_t &= \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) && \text{(Candidate cell state)}\\
C_t &= f_t * C_{t-1} + i_t * \tilde{C}_t && \text{(New cell state)}\\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) && \text{(Output gate)}\\
h_t &= o_t * \tanh(C_t) && \text{(New hidden state)}
\end{aligned}$$

- x_t: The input at time step.
- h_{t-1}: The hidden state from the previous time step (represents past memory).
- f_t: Forget gate — decides what information to discard from the previous cell state.
- i_t: Input gate — controls how much new information to store in the cell state.
- $\tilde{C}_t$: Candidate cell state — the new values that could be added to the cell state.
- C_t: Cell state — the long-term memory of the LSTM unit.
- o_t: Output gate — determines what part of the cell state is output as the hidden state.
- h_t: Hidden state — the output of the LSTM unit at time t.
- σ: Sigmoid activation function — outputs values between 0 and 1.
- tanh: Hyperbolic tangent activation function — outputs values between -1 and 1.

- W_f, W_i, W_C, W_o: Weight matrices for the forget, input, candidate, and output gates.
- b_f, b_i, b_C, b_o: Bias vectors for each corresponding gate.

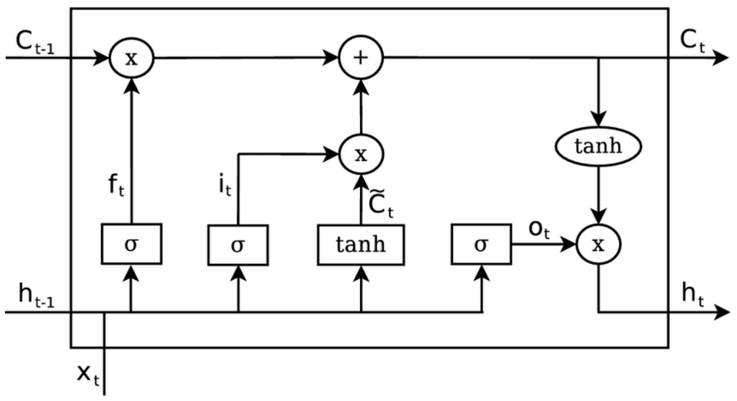


Figure 1.4: LSTM Architectures [11]

1.7 Conclusion

In this context, this study proposes the use of Generative Adversarial Networks (GANs), a class of deep learning models, to effectively address the problem of data imbalance. GANs enable the generation of highly realistic synthetic data representing rare fraudulent transactions, thereby contributing to more efficient training of deep models and enhancing their ability to distinguish between fraudulent and legitimate patterns.

This combination of generative and classification techniques strengthens the ability of fraud detection systems to adapt to emerging and evolving threats. As fraud techniques continue to develop, relying on an integrated approach that combines generative data with deep learning represents a promising direction for enhancing the security of digital financial systems and restoring user trust.

2.Related Work

Credit card fraud detection is considered one of the major research challenges in payment systems. With the rapid expansion of digital payment methods, such as e-wallets and online transactions, the need for accurate and efficient techniques to distinguish between legitimate and fraudulent transactions has become more pressing due to the massive volume of financial data and the complexity of fraud patterns.

To address this challenge, a variety of methodologies have been employed, starting with traditional statistical approaches such as anomaly detection, statistical hypothesis testing, and time series analysis, which aim to identify suspicious transactions based on numerical patterns in financial transactions. With the advancement of artificial intelligence, machine learning techniques have played a crucial role in fraud detection. Models such as logistic regression, random forests, and support vector machines (SVMs) are widely used for classifying complex financial data. Additionally, some systems rely on Naïve Bayes and Isolation Forest, which help detect anomalies and unusual patterns in financial data.These methods have limitations compared to modern models. Naïve Bayes assumes feature independence, reducing accuracy with complex data. Isolation Forests detect anomalies quickly but may fail to capture complex temporal or contextual fraud patterns.

In the field of deep learning, more advanced models have been developed, such as deep neural networks (DNNs), which utilize multiple layers to extract hidden features in transactions, and convolutional neural networks (CNNs), which are used to detect irregular patterns. Likewise, recurrent neural networks (RNNs) and long short-term memory (LSTM) networks have proven effective in analyzing sequential data and identifying temporal patterns in financial transactions. Additionally, generative adversarial networks (GANs) are employed to generate synthetic data, enhancing model accuracy by simulating potential fraudulent transactions and analyzing the differences between synthetic and real transactions. Furthermore, transformer models, which rely on attention mechanisms, are utilized to extract the most critical features from data and improve fraud detection accuracy.

This chapter aims to provide a comprehensive literature review, addressing each category of models while highlighting the strengths and weaknesses of the most effective models within each category and discussing their suitability for modern fraud detection systems.

2.1 Statistical Techniques

Statistical approaches have been extensively employed in the identification of credit card fraud, relying on the analysis of statistical properties of transaction data to detect suspicious patterns. These methods utilize thresholds or criteria to identify outlier transactions, with popular techniques including descriptive statistics, hypothesis testing, and time series analysis.

Descriptive statistics, such as mean, standard deviation, and percentiles, help uncover abnormal transactions. These metrics allow for the identification of transactions that significantly deviate from typical spending behavior, providing an initial indication of potential fraud [19].

Hypothesis testing differentiates between genuine and fraudulent transactions using statistical tests like t-tests and chi-square tests. The t-test compares the means of two groups, such as legitimate versus fraudulent transactions, to determine if there is a statistically significant difference. The chi-square test evaluates the relationship between categorical variables, such as transaction type (fraudulent or legitimate) and certain transaction characteristics, to identify patterns indicative of fraud [19].

Additionally, time series models such as ARIMA and STL provide insights into transaction patterns and trends to enhance fraud detection. ARIMA (AutoRegressive Integrated Moving Average) is widely used in forecasting time-dependent transaction behaviors, identifying unusual deviations that may signal fraud. STL (Seasonal-Trend Decomposition using Loess) helps decompose transaction data into seasonal, trend, and residual components, enabling the detection of unexpected fluctuations that may indicate fraudulent activity [32].

While these systems are somewhat effective, they lack flexibility and struggle to adapt to new types of fraud. This limitation was discussed by Bolton and Hand [6] in their study on the constraints of rule-based systems and statistical models in adapting to evolving fraud tactics.

2.2 Machine learning Techniques

2.2.1 Supervised Learning Models

This type of learning is based on pre-disaggregated data, where the model is trained in fraudulent and non-fraudulent transactions to gain the ability to distinguish them.

Decision Trees :

A decision tree is a supervised learning model used for data classification by splitting it into branches based on a set of feature-based rules [29] .The first documented use of decision trees in fraud detection dates back to the late 1990s, when they were applied to credit card fraud detection systems. One of the earliest studies in this field was conducted [**3**], who utilized decision trees alongside artificial intelligence techniques to detect fraudulent patterns in banking data. Since then, decision trees have become an essential component of fraud detection systems, widely used in modern banking and commercial applications [27].Decision trees are employed to identify unusual patterns in financial transactions by establishing rules based on historical data to differentiate between legitimate and suspicious transactions [5]. This method analyzes various features such as transaction amount, geographical location, merchant type, and user history, enabling the development of a robust model for identifying fraudulent activities based on sequential decision-making. In both the study titled "Hybrid fraud detection model

using machine learning techniques" [20] and "A supervised machine learning algorithm for detecting and predicting credit card fraud" [1] published on ScienceDirect, the Decision Tree algorithm plays a central role in identifying fraudulent transactions. The first study utilizes the Classification and Regression Tree (CART) method, where the model is trained on labeled transactional data to learn patterns distinguishing fraud from legitimate activity. The decision tree begins with a root node that selects the most informative feature for splitting the data, followed by internal decision nodes and leaf nodes that indicate the classification outcome. Similarly, the second study employs a decision tree structure that recursively partitions the data using a divide-and-conquer strategy, with each node representing a feature-based decision point. In both cases, the effectiveness of the model is assessed using standard performance metrics such as accuracy, precision, recall, and F1-score. The findings from both studies demonstrate that decision trees offer a transparent, interpretable, and accurate approach to fraud detection, highlighting their practical utility in real-world credit card fraud detection systems. Furthermore, In the study titled "A Hybrid Deep Learning Approach with Generative Adversarial

Network for Credit Card Fraud Detection" (MDPI, 2024) [23], the authors explore the integration of traditional machine learning techniques, such as Decision Trees, with deep learning methods to improve fraud detection accuracy. Although the main focus is on using Generative Adversarial Networks (GANs), Decision Trees are highlighted for their simplicity and interpretability, serving as a foundational element in the model. The hybrid approach combines the strengths of Decision Trees in structured data handling with the advanced pattern recognition capabilities of deep learning, aiming to address data imbalance and enhance real-time fraud detection performance.

Random Forests :

Is a machine learning algorithm based on ensemble learning, where it builds a robust model by combining multiple independent decision trees. Each tree is trained on a random sample of the data, and during prediction, the outputs of all trees are aggregated using majority voting (for classification) or arithmetic mean (for regression), which helps improve accuracy and reduce overfitting.

Random Forests have a strong ability to handle high-dimensional data, balance imbalanced datasets, reduce noise, and analyze feature importance, making them suitable for applications such as fraud detection, in addition to their efficiency in processing large datasets. Moreover, they outperform traditional models, such as logistic regression, in fraud detection[23][1]

The study [25] demonstrates that the Random Forest algorithm is highly effective for credit card fraud detection due to its ability to handle large, complex datasets by combining multiple decision trees. It improves detection accuracy and robustness, especially when paired with proper feature selection and data preprocessing. Overall, Random Forest proves to be a reliable and powerful tool for building efficient fraud detection systems.

In the study titled ”Credit Card Fraud Detection Using an Enhanced Random Forest Classifier for Imbalanced Data”[23], the Random Forest algorithm was used to address the problem of imbalanced data in credit card transactions. The SMOTE technique was applied to generate synthetic samples of the minority class (fraudulent transactions) to improve model performance. The results showed that the enhanced model achieved an accuracy of up to 98% and a similar F1-score, demonstrating its effectiveness in fraud detection.

These studies highlight the effectiveness of the Random Forest algorithm in fraud detection, especially when handling imbalanced datasets, and emphasize the importance of using data processing and model optimization techniques to achieve better performance in practical applications.

In one experiment,[1] the Random Forest model was applied together with undersampling to handle the issue of imbalanced data, which is a common problem in fraud datasets where legitimate transactions dominate. The results of this model showed a very high accuracy of 100%, indicating the model’s ability to correctly classify most transactions overall. However, the F1-score was relatively low (0.110), reflecting poor performance in detecting fraudulent cases specifically. This is attributed to the fact that undersampling may cause the loss of important information from the majority class, thereby negatively affecting the model’s ability to generalize fine patterns that distinguish fraud, despite achieving an apparently high accuracy.

Gradient Boosting Machines (GBM)

Is a supervised learning model that enhances predictive accuracy by iteratively combining multiple weak learners, typically decision trees, to form a robust predictive model .When researchers began applying boosting techniques to improve the detection of fraudulent credit card transactions. One of the pioneering studies in this domain was conducted by Friedman [13], who introduced gradient boosting as a powerful ensemble method capable of identifying complex fraud patterns in financial data. Since then, GBM has gained widespread adoption in fraud detection systems, particularly in banking and e-commerce applications . GBM is employed to detect fraudulent transactions by sequentially minimizing classification errors, allowing the model to focus on misclassified instances and refine decision boundaries. This approach leverages multiple decision trees, where each tree corrects the errors of the previous one, resulting in a highly accurate fraud detection system. GBM analyzes transaction attributes such as transaction amount, frequency, location, and user behavior to establish risk scores, enabling real-time fraud detection and prevention. By continuously adapting to new fraud tactics, GBM remains a powerful and widely used tool in combating financial fraud.

In the study titled ”A Hybrid Deep Learning Approach with Generative Adversarial Network for Credit Card Fraud Detection” [1], the authors proposed a hybrid model that addresses data imbalance by using Generative Adversarial Networks (GANs) to synthetically generate minority class (fraudulent) samples. This augmentation helped create a more balanced training dataset, mitigating the common issue of skewed class distributions in credit card transaction data. Once the dataset was balanced using GAN, the XGBoost algorithm was employed to classify transactions as fraudulent or legitimate. XGBoost was selected due to its high accuracy, scalability, and robustness against overfitting. The model achieved strong performance, with a precision of 0.824, recall of 0.961, F1-score of 0.887, and accuracy of 0.854. These results highlight the effectiveness of combining data augmentation via GANs with a powerful ensemble model like XGBoost for improving fraud detection in highly imbalanced environments.

In the article ”Enhanced Credit Card Fraud Detection Model Using Machine Learning” (Electronics, 2022)[3], XGBoost was utilized as one of the top-performing machine learning algorithms in a comprehensive two-stage evaluation framework. In the first stage, nine classifiers, including XGBoost, were evaluated on a real-world imbalanced credit card fraud dataset. XGBoost emerged as one of the three best models based on metrics like AUC, Recall, and F1-score. In the second stage, the authors applied 19 different resampling techniques—such as oversampling, undersampling, and hybrid approaches—to further improve model performance. Notably, XGBoost paired with the Borderline-SMOTE technique achieved strong results, including an AUC of 96.47%, Recall of 92.98%, Precision of 82.31%, and F1-score of 87.30%. This underscores XGBoost’s capability to handle class imbalance effectively and detect fraud with high accuracy, especially when enhanced by appropriate sampling methods.

In the study “Enhanced Credit Card Fraud Detection Model Using Machine Learning” (Electronics, 2022)[3],, the authors implemented a hybrid approach that combines XGBoost with SVMSMOTE to address class imbalance in credit card fraud detection. SVMSMOTE enhances the traditional SMOTE method by generating synthetic minority samples near the support vectors, focusing on areas where class overlap is most likely to occur. This improved sampling strategy helps the classifier better learn the boundary between fraudulent and legitimate transactions. When paired with the XGBoost classifier, this method achieved strong performance results: precision of 0.808, recall of 0.930, and an F1-score of 0.865. These results highlight the effectiveness of using SVMSMOTE with ensemble methods like XGBoost to improve sensitivity and overall classification quality in imbalanced fraud datasets.

In credit card fraud detection, data typically suffers from an imbalance problem, where fraudulent transactions constitute a very small percentage compared to normal transactions. This creates significant challenges when training machine learning models. This disparity causes models to be biased toward the majority class (normal transactions), as they may learn to classify all transactions as ”normal” to achieve superficially high accuracy, while failing to detect fraud. Therefore, addressing data imbalance in credit card transactions is essential to ensure the accuracy and effectiveness of fraud detection systems. All previous methods, such as machine learning algorithms and classification techniques, will not be effective unless the data imbalance problem is properly corrected. Hence, we explore the following methods to solve this issue.

Synthetic Minority Over-Sampling Technique (SMOTE)[7]is a machine learning algorithm designed to address the issue of data imbalance by generating new synthetic samples for the underrepresented class instead of simply duplicating the original data. SMOTE works by selecting samples from the minority class, identifying their nearest neighbors using the K-Nearest Neighbors (KNN) algorithm, and creating new data points between the original sample and its closest neighbors, thereby enhancing the representation of the minority class. Despite its effectiveness, SMOTE can introduce noisy or misleading data that negatively impacts model performance. To overcome this, SMOTE + Edited Nearest Neighbors(SMOTE-ENN) was developed, combining synthetic data generation with the removal of irrelevant or noisy samples after oversampling. This refinement improves data quality and prevents excessive noise accumulation.

Generative Adversarial Networks (GANs) [12]are a deep learning technique used to generate synthetic data that closely mimics real data, making them an effective tool for addressing data imbalance in fraud detection. GANs consist of two competing models: the Generator, which creates new data based on patterns in real data, and the Discriminator, which attempts to distinguish between real and generated data. Through continuous competition, the Generator progressively improves its performance, producing realistic fraudulent data that is difficult to differentiate from actual data. This enhances the accuracy and efficiency of fraud detection models, particularly when genuine fraudulent data is scarce.

2.2.2 Unsupervised Learning Models

These models are used when tagged data is not available, as they rely on the detection of anomalies without the need for prior knowledge of fraud cases

Anomaly detection

Anomaly detection is an effective approach for credit card fraud detection, as it focuses on identifying transactions that deviate from normal patterns without requiring a large amount of labeled data. It is typically implemented through two main methods: distance-based approaches, which flag data points that are far from the overall distribution, and density-based approaches, which identify anomalies in low-density regions. Studies have demonstrated the effectiveness of these methods in detecting fraud, especially in imbalanced datasets, making anomaly detection a valuable tool in financial systems.[6].

One-Class SVM

One-Class Support Vector Machine (One-Class SVM) is an effective method for anomaly detection, as it creates a boundary that separates normal and abnormal transactions based on data distribution. This model learns the pattern of normal data and classifies any point that falls outside this pattern as an anomaly. One-Class SVM is particularly useful in cases where labeled data is limited, as it does not require a large amount of fraudulent data for training, making it suitable for credit card fraud detection.

Isolation Forest

Isolation Forest is another powerful technique for anomaly detection, relying on repeatedly partitioning the data into random decision trees. Anomalies are easily identified as they get isolated early in the partitioning process, whereas normal data points require more splits to reach complete isolation. Isolation Forest is highly efficient in handling large and imbalanced datasets, making it an ideal choice for detecting rare fraudulent transactions in financial systems[30].

In this study ”Anomaly Detection in Credit Card Fraud” [30], features were standardized before training to ensure each feature contributes equally to the splits. The fraction of expected anomalies (contamination parameter) was set based on the approximate proportion of fraudulent transactions in the dataset to help the model set an appropriate threshold for flagging anomalies.

The model’s ability to identify 90% of fraudulent transactions (high recall) shows its strength in isolating true frauds. However, the moderate precision (46%) indicates many normal transactions were incorrectly flagged, suggesting the need for further tuning or combination with other techniques to reduce false positives.

K-Means

Is an unsupervised learning algorithm used to cluster data into K similar groups based on shared characteristics. The algorithm initializes K random centroids and assigns each data point to the nearest centroid based on Euclidean distance. The centroids are then updated based on the average of the points within each cluster, and the process repeats until the centroids stabilize. This algorithm is used in fraud detection by classifying unusual transactions into separate clusters, where points far from any main cluster can be considered suspicious[21].

This work ”Credit Card Fraud Detection Using Spectral Clustering”[21] provides a comprehensive overview of anomaly detection with a focus on credit card fraud detection using spectral clustering. It begins by explaining the fundamentals of anomaly detection, covering its types—such as point, contextual, and collective anomalies—and various detection algorithms including statistical, machine learning, proximity-based, and ensemble methods. The tutorial then introduces spectral clustering, contrasting it with traditional clustering techniques like K-Means, and explaining its advantages in capturing complex data structures. It details the methodology of spectral clustering, including how to select the optimal number of clusters. Finally, the tutorial demonstrates how to apply spectral clustering to detect fraudulent credit card transactions through proper data preparation and model implementation. The model achieved a precision of 0.730, recall of 0.880, and an F1-score of 0.790, indicating strong performance in accurately identifying fraudulent transactions while maintaining a good balance between false positives and false negatives.

Overall, it illustrates how spectral clustering can effectively identify anomalies to enhance fraud detection and improve transaction security.

2.3 Deep Learning Techniques

Convolutional Neural Networks (CNNs) are a type of deep neural networks designed to extract spatial features from data through convolutional layers and pooling layers. Although CNNs were originally developed for processing visual data, they have been adapted for financial fraud detection by representing transaction data in a Grid Representation, allowing for the analysis of spatial patterns within it. Grid Representation refers to a method of organizing data into a two-dimensional or multi-dimensional matrix, where numerical values are arranged in a grid-like structure. This enables the model to understand spatial and temporal relationships between different features, thereby enhancing the accuracy of fraud detection.

Recurrent Neural Networks (RNNs) are a type of neural network designed to process sequential data by maintaining a hidden state, allowing them to learn temporal dependencies. However, traditional RNNs suffer from the vanishing gradient problem, which limits their ability to capture long-range dependencies.

To address this issue, Long Short-Term Memory (LSTM) networks were developed as an advanced extension of RNNs. LSTMs utilize gated memory units to regulate the flow of information and control the retention of important data over extended periods.

In fraud detection, RNNs and LSTMs are used to analyze time series of financial transactions, enabling them to learn customer behavioral patterns and identify anomalies. By capturing temporal dependencies, LSTM models can distinguish between normal and suspicious activities, improving fraud detection accuracy compared to traditional methods.

Transformers are a neural network architecture based on the Self-Attention Mechanism and Multi-Head Attention, allowing them to process sequential data in parallel rather than using the iterative approach of (RNNs).Transformers excel at capturing long-range dependencies in data by distributing attention across all elements of the input sequence simultaneously.

In fraud detection, Transformer-based models leverage these capabilities to analyze streaming payment transactions, effectively identifying complex interactions between financial transactions and detecting anomalous patterns with high efficiency. The self-attention mechanism enables the model to focus on the most relevant transactions, enhancing fraud prediction accuracy compared to traditional RNN-based models.

The study titled ”A Soft Voting Ensemble Learning Approach for Credit Card Fraud Detection”[24] presents a method that combines multiple classifiers using a soft voting ensemble technique to enhance fraud detection accuracy. To address the prevalent issue of class imbalance in credit card transaction datasets, the researchers employed various sampling techniques, including oversampling, undersampling, and hybrid methods. Among these, the Synthetic Minority Over-sampling Technique (SMOTE) was utilized to generate synthetic examples of the minority class (fraudulent transactions), thereby improving the model’s ability to learn from limited fraudulent data.

The ensemble model integrates several classifiers, notably the Multi-Layer Perceptron (MLP), which is adept at capturing complex nonlinear patterns in data. By aggregating the predictions of individual classifiers through soft voting, the ensemble approach assigns probabilities to each class and selects the class with the highest combined probability. This method enhances the model's robustness and generalization capabilities.

2.4 Hybrid Deep Learning Ensemble

In the study "A Hybrid Deep Learning Ensemble Model for Credit Card Fraud Detection,"[16] the researchers proposed a hybrid model combining deep learning and machine learning techniques to enhance the accuracy of fraud detection in credit card transactions. The model was designed using three advanced neural networks: Convolutional Neural Networks (CNNs) for spatial feature extraction, Long Short-Term Memory (LSTM) networks for capturing temporal patterns in transactions, and Transformer models to leverage self-attention mechanisms for analyzing complex relationships between transactions. After extracting features from each of these models, their outputs were combined using XGBoost as a meta-learner, allowing for improved prediction accuracy and reduced false positive rates

The issue of class imbalance was not addressed using resampling techniques like SMOTE. Instead, the researchers relied on the deep learning models' ability to learn advanced representations directly from raw data without artificially balancing the dataset. The model was tested on two different datasets (a European dataset and a Taiwanese dataset), where the results showed that the proposed approach achieved significant improvements in recall and overall accuracy compared to traditional models such as decision trees and logistic regression

In the study "A Hybrid Deep Learning Approach with Generative Adversarial Network for Credit Card Fraud Detection," [23]the researchers proposed a hybrid model that combines Recurrent Neural Networks (RNNs) and Generative Adversarial Networks (GANs) to address the challenge of fraud detection in financial transactions, particularly in dealing with class imbalance, where fraudulent transactions are significantly fewer than legitimate ones. The study leveraged GANs to generate synthetic fraudulent samples that mimic real-world patterns, helping to enhance the model's ability to distinguish between legitimate and suspicious transactions. In this framework, the generator was responsible for creating new samples representing potential fraudulent transactions, while the discriminator, based on LSTM and GRU networks, was trained to classify both original and generated transactions.

The proposed model relied on analyzing the temporal nature of financial transactions, focusing on recurring behavioral patterns that may indicate fraud. Compared to traditional models, this approach offers a greater ability to detect fraudulent activities that occur over extended periods or involve subtle changes in user behavior. Furthermore, the study highlighted that combining deep recurrent learning with intelligent data generation techniques can contribute to developing more robust and effective fraud detection systems.

2.5 Comparison Of State-Of-The-Art Techniques in Credit Card Fraud Detection

To provide insight into the ovrall reserch lanscape,we summarize pervious related work in Table 2.1, which outlines some key specifications of the aforementioned techniques (e.g., Techniques Used, results).

Reference	Method	Precision	Recall	F1-Score
Madhurya and Gur[20]	Decision Tree	-	-	0.369
Afriyie and Tawi [1]	Decision Tree	0.050	0.930	0.090
Mienye and Swart [23]	Decision Tree	0.780	0.570	0.660
Mrozek and al. [25]	Random forest + undersampling	-	1.00	0.110
Mienye and Swart [23]	Random forest	0.864	0.802	0.832
Afriyie and Tawi [1]	Random forest	0.090	0.970	0.170
Mienye and Swart [23]	XGBoost	0.887	0.824	0.854
Alfaiz and Fati [3]	XGBoost	0.800	0.952	0.869
Alfaiz and Fati [3]	XGBoost + SVMSMOTE	0.808	0.930	0.865
Alarfaj and Malik [2]	Decision Tree	-	-	0.810
33rd Square [30]	Isolation-Forest	0.460	0.900	0.610
Mangla, P. [21]	Spectral Clustering	0.730	0.880	0.790
Kasasbeh and Alda [18]	MLP	0.802	0.905	0.851
Majadi and Mazu [24]	MLP + SMOTE	0.098	0.928	0.177
Mienye and Swart [23]	GAN + GRU	1	0.992	0.996
Mienye and Swart [23]	GRU	0.850	0.897	0.873
Mienye and Swart [23]	GAN + LSTM	0.979	0.990	0.984
Mienye and Swart [23]	GAN + Simple RNN	0.962	0.960	0.961
Mienye and Jere [22]	CNN	0.864	0.714	0.780
Ileberi and Sun [16]	CNN	0.784	0.628	0.697
Alarfaj and Malik [2]	CNN	0.932	0.775	0.846
Ileberi and Sun [16]	Transformer	0.890	0.720	0.797
Ileberi and Sun [16]	XGBoost based Stacked Ensemble DL	0.989	0.961	0.975

Table 2.1: Comparison of studies using the European Credit Card Dataset

2.6 Conclusion

At the end of this chapter, we conclude that each approach has its strengths and limitations. Traditional models struggle to adapt to evolving fraud patterns, while deep learning techniques offer higher accuracy but require significant computational resources and large amounts of data.

Despite advancements in fraud detection, fraudulent techniques continue to evolve, making some methods ineffective against new, previously unseen fraud strategies. Additionally, class imbalance remains a major challenge, as fraudulent transactions are rare, making it difficult to achieve effective data balancing, even when employing techniques such as SMOTE or GANs, which may not fully capture the complexity of real-world fraud patterns.

Based on the findings of this chapter, we aim to develop hybrid models that integrate traditional and deep learning techniques to enhance accuracy and adaptability, addressing the evolving nature of fraudulent strategies. This will be explored further in the next chapter.

3.Materials and Methods

3.1 Introduction

Class imbalance is one of the most significant challenges in the field of credit card fraud detection. To address this issue, a collection of benchmark datasets—each suffering from severe class imbalance—was utilized. Initially, the Wersstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) was employed to balance these datasets. Subsequently, it was integrated with Fuzzy Logic using three different strategies: applying it to the generator only, to the discriminator only, and to both the generator and the discriminator simultaneously. These configurations aimed to enhance the quality of data generation and improve discrimination capability under uncertainty and ambiguity. Afterwards, several models (LSTM, CNN, and XG-Boost) were trained on the balanced data to more effectively learn the temporal and spatial patterns associated with fraudulent behavior.

3.2 Uniform Manifold Approximation and Projection

UMAP Uniform Manifold Approximation and Projection is a modern dimensionality reduction technique designed to simplify high-dimensional data while preserving its internal structure. UMAP is faster and more efficient than methods like t-SNE and tends to maintain both local and global relationships between samples. It is widely used for data visualization, pattern discovery, and complex data analysis, making it an excellent choice for tasks like fraud detection and classification analysis.

we used clustering to divide the normal transaction data into 10 groups (clusters), then selected an equal number of samples from each group based on the number of fraudulent transactions in the dataset. The purpose of this step is to ensure a balanced and comprehensive representation of the various patterns of normal transactions, so that the final sample includes all types of normal data from different groups. After that, I combined the selected samples with all the fraudulent transactions to form a balanced dataset.

3.3 Datasets

In this study, three datasets were used, as they are among the most well-known. The first is the European credit card dataset,the second is PaySim Synthetic Mobile-Money dataset ,and third dataset is The Synthetic Credit Card Transactions . These datasets are characterized by a significant imbalance in transaction distribution, with fraudulent cases being extremely rare. This imbalance poses a significant challenge in learning and fraud detection.

3.3.1 European Credit Card Dataset

This European credit card dataset, gathered in September 2013, is a popular standard for studying financial fraud detection systems. It contains 284,807 credit card transactions made by European cardholders over two days, with only 492 transactions marked as fraudulent. This leads to a strongly imbalanced dataset; fraudulent transactions represent only 0.172 % of the total. This uneven distribution presents a problem for machine learning algorithms, as models trained on this data often favor the majority class— legitimate transactions — and struggle to accurately spot the infrequent fraudulent ones.

To counter this, researchers and professionals have tried different data balancing methods to enhance model performance. Common methods include undersampling the majority class (by randomly decreasing the number of legitimate transactions), oversampling the minority class (by duplicating fraudulent samples), and more complex methods like SMOTE (Synthetic Minority Over-sampling Technique), which creates synthetic fraudulent samples based on existing ones. These methods aim to create a more balanced training set, enabling machine learning models to learn distinct patterns efficiently without being overwhelmed by the majority class. The dataset, featuring 30 features, including anonymized PCA-transformed components (V1 to V28), along with ’Time’ and ’Amount’, offers a useful environment for experimenting with these methods and assessing how well fraud detection algorithms work in real-world situations.

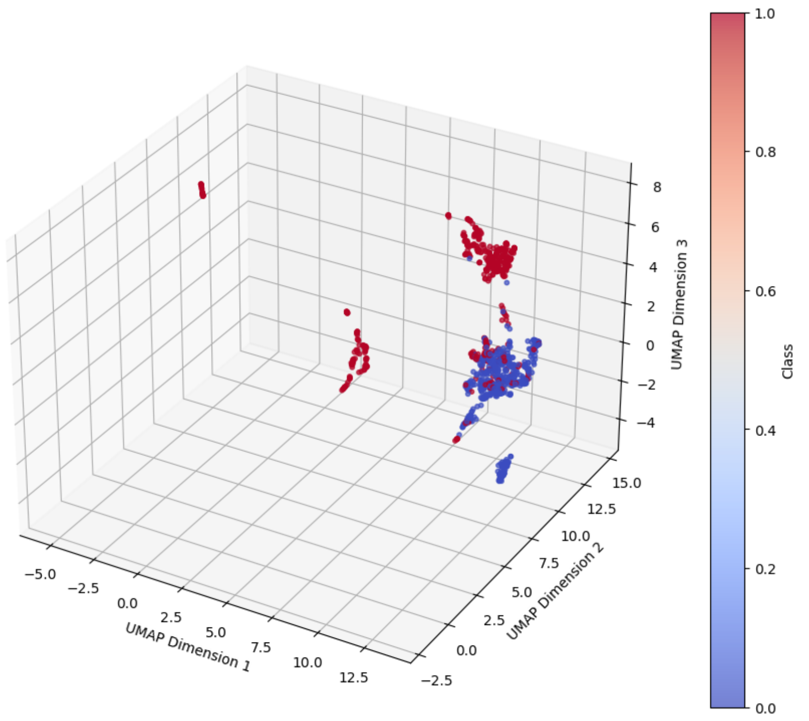


Figure 3.1: UMAP of European dataset

3.3.2 PaySim Synthetic Mobile-Money Transactions

PaySim is considered one of the most prominent synthetic datasets used in the field of financial fraud detection. It was released in 2016 by the researcher E.A. Laxmi through the Kaggle platform. This dataset was developed by simulating real mobile money transactions, with the aim of overcoming restrictions related to the confidentiality of financial data, especially in developing countries. It contains approximately 6,362,620 transactions, each labeled as either fraudulent or non-fraudulent using the isFraud feature. The proportion of fraudulent transactions is around 0.129%, corresponding to 8,213 instances, making it a realistic representation of the class imbalance problem commonly found in fraud detection tasks.

The dataset covers a variety of transaction types, including cash-in, cash-out, transfers, payments, and debits. Each transaction is characterized by a set of features that describe its properties. These include the step, which indicates the time sequence of the transaction; the type of transaction; the amount involved; the origin account ID (nameOrig); the account balance before and after the transaction (oldbalanceOrg and newbalanceOrig); the destination account ID (nameDest); and the balance of the receiving account before and after the transaction (oldbalanceDest and newbalanceDest). Additionally, the dataset includes two key indicators: isFraud, which identifies whether the transaction was indeed fraudulent, and isFlaggedFraud, which shows whether the system flagged the transaction as suspicious.

Due to its realistic structure and the diversity of transaction behaviors it represents, the PaySim dataset has been widely adopted in academic research focused on fraud detection using artificial intelligence and machine learning techniques, particularly in scenarios dealing with imbalanced data distributions.

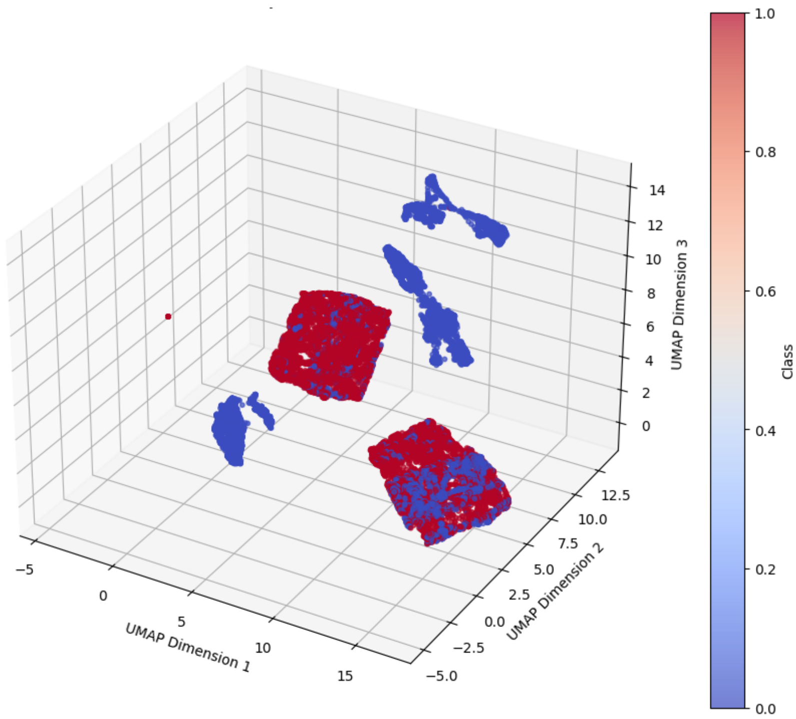


Figure 3.2: UMAP of pysim dataset

3.3.3 The Synthetic Credit Card Transactions

The Credit Card Transactions Fraud Detection Dataset is a synthetic dataset created to simulate real-world credit card transaction patterns, with the aim of developing and testing fraud detection systems. The data was generated using Sparkov, a simulator designed to produce realistic sequences of financial transactions. These transactions represent simulated activity over the period from January 1, 2019, to December 31, 2020.

The dataset includes 21 features along with a target column (class) that indicates whether a transaction is fraudulent or normal. Each transaction contains information such as the transaction time, amount, and several anonymized variables that mimic customer behavior. The dataset consists of a total of 1,852,397 transactions, divided into a training set with 1,296,675 records and a test set with 555,719 records. Among these, 9,651 transactions are labeled as fraudulent, while 1,812,743 are labeled as normal, reflecting the class imbalance typically observed in real-world financial fraud datasets.

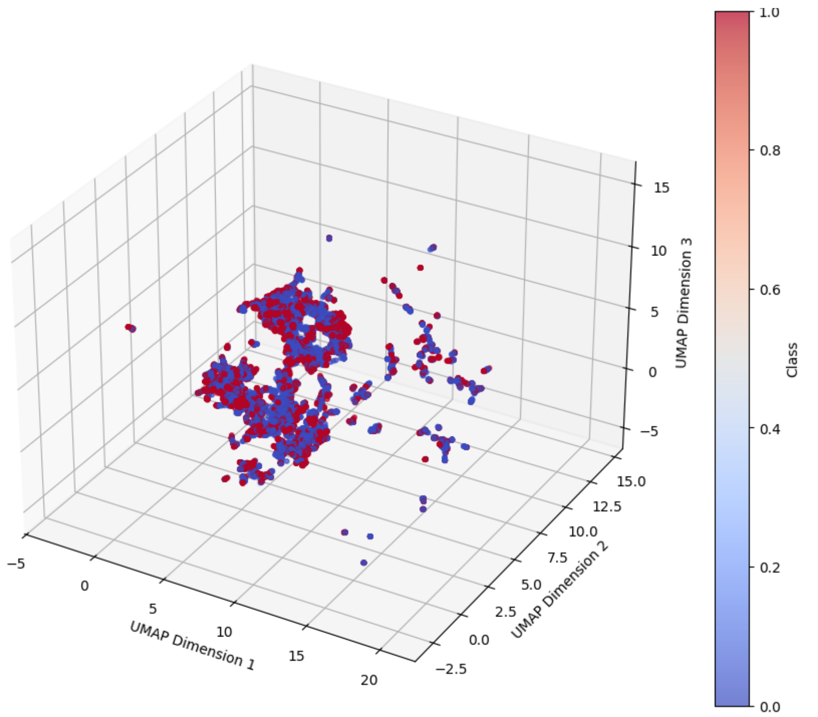


Figure 3.3: UMAP of the synthetic dataset

3.3.4 Overview of the datasets.

Dataset	Samples	Fraudulent	Normal	Features	Type
European dataset	284,807	492	284,315	30	Numeric
PaySim dataset	6,362,620	8,213	6,354,407	10	(Numeric & Categorical)
Synthetic dataset	1,852,397	9,651	1,812,743	21	(Numeric & Categorical)

Table 3.1: Summary of the credit card datasets.

3.4 Proposed Method

3.4.1 WGAN-GP

1. Generator

The generator is a neural network designed to transform random noise (latent vectors) into synthetic data that resembles real credit card transaction features. It achieves this through a series of fully connected layers, utilizing activation functions such as LeakyReLU and tanh to ensure the generated data falls within the same range as the real data. Batch normalization is applied to stabilize training and improve convergence. The generator's goal is to learn to produce realistic samples that are indistinguishable from genuine transactions by the discriminator.

The Generator architecture is summrized in Table 3.2 as follows :

Component	**Description**
`make_generator`	Builds a simple feedforward generator model for GANs.
`Dense Layers`	Project the latent input into higher-dimensional representations.
`LeakyReLU Activations`	Allow gradient flow for small negative values to prevent dead neurons.
`BatchNormalization`	Stabilizes training by normalizing layer outputs, accelerating convergence.
`Output Layer (tanh)`	Produces the final synthetic output in the range $[-1, 1]$, matching typical GAN output normalization.

Table 3.2: Generator neural network architecture

2. Discriminator

The discriminator is a neural model used in Generative Adversarial Networks (GANs) to distinguish between real and fake data. In this work, `SpectralDense` layers are used to constrain weight values and make the training more stable. `LeakyReLU` activation functions are also applied to facilitate learning, especially when dealing with negative values. The final layer outputs a single value used to evaluate whether the sample is real or generated.

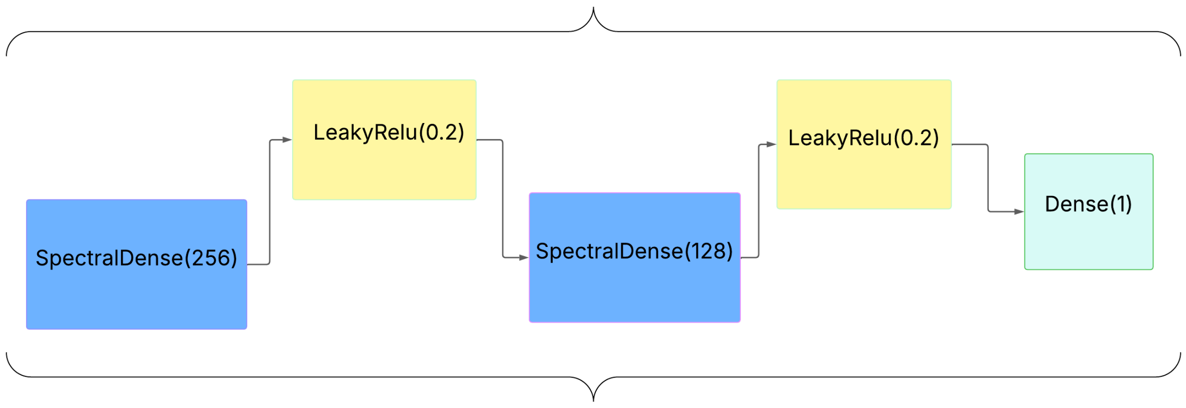


Figure 3.4: Discriminator Neural Network

Components	Description
`SpectralDense(units)`	Custom dense layer that applies spectral normalization to the weights before forward pass.
`self.units = units`	Stores the number of output units for the dense layer.
`layers.Dense(..., kernel_initializer="he_normal")`	Initializes the dense layer weights using He initialization for better performance.
`self.u = self.add_weight(...)`	Adds a non-trainable weight vector used in power iteration to approximate spectral norm.
`tf.reshape(w, [-1, w_shape[-1]])`	Flattens the kernel weight to a 2D matrix for spectral normalization.
`tf.linalg.l2_normalize(...)`	Normalizes vectors during the power iteration process.
`w_bar = w_reshaped / sigma`	Normalized weight matrix using the estimated spectral norm σ.
`self.dense.kernel.assign(w_bar)`	Assigns the normalized weights back to the Dense layer.
`return self.dense(inputs)`	Applies the normalized Dense layer to the input.

Table 3.3: SpectralDense Layer Components

3. Gradient Penalty

To enforce the Lipschitz continuity constraint in GANs, a **gradient penalty** is added to the loss function instead of weight clipping. This penalty encourages the gradient norm of the critic with respect to the input to be close to 1.

Given:

- Real samples: x_{real}
- Generated (fake) samples: x_{fake}
- Interpolation factor: $\alpha \sim \mathcal{U}(0, 1)$

We compute the interpolated samples:

$$\hat{x} = \alpha x_{\text{real}} + (1 - \alpha) x_{\text{fake}}$$

Let $D(\cdot)$ be the critic. The gradient penalty is given by:

$$\text{GP} = \mathbb{E}_{\hat{x}} \left[\left(\|\nabla_{\hat{x}} D(\hat{x})\|_2 - 1 \right)^2 \right] \tag{3.1}$$

This term is added to the critic loss to penalize gradients that deviate from norm 1.
Why Gradient Penalty?

- It stabilizes GAN training.
- Prevents exploding or vanishing gradients.
- Encourages smooth and realistic generation of synthetic data.

3.4.2 Fuzzy Logic

To enhance the GAN's capability of handling uncertainty and imprecise data patterns, we designed a custom **FuzzyLayer** to integrate fuzzy logic into the network. This layer introduces interpretable reasoning mechanisms that are particularly useful in real-world problems such as credit card fraud detection, where data ambiguity and overlapping patterns are common.[26]

Given an input feature vector $x \in \mathbb{R}^N$, we partition it into four segments: $a \in \mathbb{R}^j, b \in \mathbb{R}^k, c \in \mathbb{R}^l, d \in \mathbb{R}^m$, with $j + k + l + m = N$. These segments represent fuzzy antecedents for rule-based computation.

- **T-norm (Fuzzy AND)**:
  The T-norm models the fuzzy conjunction, capturing the degree to which all antecedents are simultaneously satisfied. The formula

  $$T = a \circ \prod b$$

  applies element-wise multiplication of vector a with the product of elements in segment b. This enforces a strict co-activation requirement: all relevant antecedents must have strong positive contributions for the fuzzy rule to activate. Using the product operator on b means that if any element in b is low or zero, the overall conjunction value T is greatly diminished, reflecting the logical AND's nature of requiring all conditions to hold. This strictness helps reduce false positives by ensuring only strong joint evidence triggers rule activation, which is crucial for tasks like fraud detection where precision is key.

- **S-norm (Fuzzy OR)**:
  The S-norm models fuzzy disjunction, representing the degree to which at least one antecedent holds true. The probabilistic sum formulation

  $$S = c + d - c \circ d$$

  allows combining inputs flexibly without exceeding a total of 1, making it well-suited for overlapping or noisy data. Taking the product

  $$S_{\text{prod}} = \prod S$$

  aggregates multiple fuzzy OR contributions, allowing rules to be activated even if only some antecedents are partially true. This flexibility enhances robustness against partial truths and noisy inputs, common in real-world ambiguous datasets.

- **Interpolative Rule Combination (IRC)**:
  Combining the strict T-norm and flexible S-norm outputs, the IRC formula:

  $$IRC = 1 - T + T \circ S_{\text{prod}}$$

balances strict and flexible inference by interpolating between AND and OR behaviors. When T is low, IRC tends toward $1 - T$, reflecting weak conjunction activation. When T is high, IRC depends more on the aggregated OR term S_{prod}, capturing the flexibility of partial antecedent satisfaction. This interpolation mimics complex human-like fuzzy reasoning, where rules are neither fully rigid nor completely loose.

- **Sigmoid Smoothing and Scaling**:
  To convert the IRC score into a normalized confidence score within $[0, 1]$, a scaled sigmoid transformation is applied:

$$\text{fuzzy\_score} = \frac{(1 + e^{4.5}) \cdot \sigma(9 \cdot (IRC - 0.5)) - 1}{e^{4.5} - 1}$$

  where

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

  is the sigmoid function. The constants 4.5 and 9 were chosen empirically to produce smooth gradients and sharp transitions around decision boundaries (IRC near 0.5), which facilitates stable and effective model training. This smoothing prevents abrupt cutoff effects, allowing the network to learn nuanced fuzzy boundaries instead of hard thresholds.

- **Final Output**:
  The overall fuzzy confidence score is obtained by multiplying all individual fuzzy scores:

$$\text{confidence} = \prod \text{fuzzy\_score}$$

  This multiplicative aggregation ensures that the final confidence is high only if all fuzzy rules are strongly activated simultaneously. This reflects the intuition in fraud detection that multiple indicators must align to confidently flag suspicious behavior, reducing false alarms from isolated weak signals.

Integrating these fuzzy operations provides interpretability, handles partial truths effectively, and softens crisp decision boundaries—key traits for fraud detection tasks where data is often ambiguous or incomplete.

Method 1: Discriminator with Fuzzy logic

In the first method, a Generative Adversarial Network (GAN) is enhanced by incorporating both fuzzy logic and spectral normalization techniques. The **Generator** network starts with dense layers followed by LeakyReLU activations and BatchNormalization to ensure non-linearity and stable training. The generator creates synthetic data, applying a `tanh` activation at the output layer, which ensures the generated data is within the $[-1, 1]$ range.

The **Discriminator** combines both `SpectralDense` layers and a `FuzzyLayer`. The `SpectralDense` layers use spectral normalization to stabilize the training of the model, while the `FuzzyLayer` introduces fuzzy logic into the decision-making process, mimicking uncertainty and imprecision in credit card transaction data.

This hybrid model allows the discriminator to better handle ambiguous features, improving the model's ability to differentiate between real and synthetic data. The combination of spectral normalization and fuzzy logic introduces a novel approach to improving GAN performance in tasks like fraud detection, where data patterns can be noisy or uncertain.

Method 2: Generator with Fuzzy logic

In the second method, the GAN structure is slightly modified by integrating fuzzy logic directly into the generator. The **Generator** consists of fully connected layers, LeakyReLU activations, and BatchNormalization, similar to the first method, but with an added `FuzzyLayer`.

The generator uses the fuzzy logic layer to modulate the generated data by incorporating fuzzy scores derived from the inputs. The `FuzzyLayer` computes fuzzy membership and logical operations, producing outputs that modulate the generator's results. The final output is generated through a `tanh` activation, modulated by the fuzzy score.

The **Discriminator** here is a simpler version, consisting of dense layers followed by LeakyReLU activations. This setup allows the generator to produce more diverse synthetic data, and the discriminator to assess the data more effectively by incorporating fuzzy logic into its decision process, making it more adaptive to the uncertainties present in fraud detection tasks.

Method 3: Full Integration of Fuzzy Logic in Generator and Discriminator

The third method fully integrates fuzzy logic into both the **Generator** and **Discriminator**. The generator in this setup incorporates fuzzy logic not just in the output layer but throughout its architecture. It starts with fully connected layers and applies fuzzy logic modulation using the `FuzzyLayer` at the output stage.

The discriminator similarly uses a feature extraction network followed by fuzzy logic via the `FuzzyLayer`. Both networks use `sigmoid` activations in the fuzzy logic components to integrate uncertainty into the decision-making process. The fuzzy logic layers enable the GAN to handle the imprecise nature of transaction data, which is typical in fraud detection tasks.

This architecture improves the generator's ability to produce realistic data and enhances the discriminator's ability to assess data with ambiguous characteristics. The full integration of fuzzy logic in both the generator and discriminator enables more nuanced and effective fraud detection, particularly when dealing with noisy or uncertain credit card transaction data.

3.4.3 Hyperparameter Tuning

We leverage **Optuna** library to automate hyperparameter tuning for our **WGAN-GP**. We define an `objective` function that, on each trial, samples values for `latent_dim`, `learning_rate`, `n_critic` and `batch_size`, trains the GAN on real fraud vs. non-fraud data, computes the Fréchet Inception Distance (FID) between real and generated fraud samples, and returns the FID for Optuna to minimize. By running a fixed number of trials (e.g., 20), Optuna efficiently explores the search space and pinpoints the hyperparameter set that produces the lowest FID, thus yielding the most realistic synthetic fraud data.

Hyperparameter	**Type**	**Search Space**	**Description**
`latent_dim`	Integer	16–128	Dimensionality of the noise input to the generator.
`learning_rate`	Float (log)	10^{-4}–10^{-3}	Learning rate for both generator and discriminator optimizers.
`n_critic`	Integer	3–10	Number of discriminator updates per generator update (GAN training).
`batch_size`	Integer	{32, 64, 128}	Number of samples per training batch.
`Fuzzy_dim`	Integer	{8, 16, 24}	Size of the input segment used by the fuzzy logic layer.

Table 3.4: Optuna Hyperparameters for WGAN-GP Tuning

Algorithm 1 Synthetic Fraud Generation Using WGAN-GP with Optional Fuzzy Logic

```
1:  Input: Dataset, Method $\in \{1, 2, 3, 4\}$
2:  Normalize features to range $[-1, 1]$
3:  Separate fraud and non-fraud transactions
4:  Define Fuzzy Modes:
5:  if Method == 1 then
6:      No fuzzy logic in Generator or Discriminator
7:  else if Method == 2 then
8:      Apply fuzzy logic in Discriminator only
9:  else if Method == 3 then
10:     Apply fuzzy logic in Generator only
11: else if Method == 4 then
12:     Apply fuzzy logic in both Generator and Discriminator
13: end if
14: Define Objective Function for Optuna:
15: for each trial do
16:     Sample hyperparameters: latent_dim, learning_rate, batch_size, n_critic
17:     Initialize Generator $G$ and Discriminator $D$
18:     if Method == 2 or Method == 4 then
19:         Add FuzzyLayer to $D$
20:     else
21:         Set $D.fuzzy$ = None
22:     end if
23:     if Method == 3 or Method == 4 then
24:         Add FuzzyLayer to $G$
25:     else
26:         Set $G.fuzzy$ = None
27:     end if
28:     for epoch = 1 to max_epochs do
29:         for $k = 1$ to $n\_critic$ do
30:             Sample real fraud batch $x_{\text{real}}$
31:             Sample noise $z \sim \mathcal{N}(0, I)$
32:             Generate fake batch $x_{\text{fake}} = G(z)$
33:             Interpolate $x_{\text{interp}} = \alpha x_{\text{real}} + (1 - \alpha)x_{\text{fake}}$
34:             Compute gradient penalty: $GP = (\|\nabla D(x_{\text{interp}})\|_2 - 1)^2$
35:             Discriminator loss: $D_{\text{loss}} = D(x_{\text{fake}}) - D(x_{\text{real}}) + \lambda \cdot GP$
36:             Update $D$
37:         end for
38:         Sample new noise $z$
39:         Generate fake batch $x_{\text{fake}} = G(z)$
40:         Generator loss: $G_{\text{loss}} = -D(G(z))$
41:         Update $G$
42:     end for
43:     Generate synthetic fraud samples from trained $G$
44:     Evaluate:
            • FID (Fréchet Inception Distance)
45:     Return FID to Optuna
46: end for
47: Select the best generator $G^*$ with the lowest FID and display the balanced dataset.
```

3.4.4 LSTM Module

Define a sequential LSTM-based deep learning model designed to process fraud detection by sequential data.This structure allows the model to learn from multiple observations over time, with multiple feature dimensions.

Module Architecture

1. **First LSTM Layer :** This layer reads the transaction sequence data over time. It captures temporal dependencies between transactions by processing the input as a series of time steps (M steps, each with N features). The output is a sequence of hidden states of shape $(M, 50)$, allowing the model to retain detailed temporal patterns which might indicate fraudulent behavior.

2. **Second LSTM Layer :** Takes the sequence from the previous layer and outputs only the last hidden state of shape $(100,)$. This step condenses the learned temporal information into a fixed-size representation that summarizes the entire transaction sequence, helping the model detect patterns typical of fraud across the full timeline.

3. **Dropout Layer :** This layer prevents the model from overfitting by randomly disabling 50% of the neurons during training. This improves the model's ability to generalize to unseen fraudulent and non-fraudulent sequences.

4. **Dense Output Layer :**

   Outputs a single probability value between 0 and 1, representing the likelihood of the input transaction sequence being fraudulent. If the value is closer to 1, the transaction is predicted as fraud; otherwise, it is classified as normal.

3.4.5 CNN Module

Define a sequential Convolutional Neural Network (CNN)-based deep learning model designed for fraud detection using image-like data representations. The model is effective in identifying patterns in transaction data that may indicate fraudulent activity.

Module Architecture

1. **First Convolutional Layer :** This layer applies 32 filters of size (3×3) to the input data (an $N \times M$ matrix with 1 channel). These filters extract low-level features such as edges, textures, or simple local patterns, forming the foundation for recognizing subtle combinations of characteristics indicative of fraudulent behavior. The ReLU activation function is then applied to the convolution outputs to introduce non-linearity, enabling the model to learn more complex patterns and relationships in the data.

2. **Max-Pooling Layer :** This layer performs max-pooling with a (2×2) window, reducing the spatial dimensions of the feature map. It helps decrease computational complexity, focus on the most relevant features, and prevent overfitting by downsampling the feature map.

3. **Flatten Layer :** This layer flattens the pooled feature map into a one-dimensional vector, preparing it for the fully connected layers. It reshapes the feature map into a format that can be processed for classification.

Dense Layer : A fully connected layer with 64 units that learns higher-level features from the flattened data. The ReLU activation function allows the model to capture more complex, non-linear relationships in the data.

Dense Output Layer : The final layer consists of a single output unit that uses the Sigmoid activation function to output a probability value between 0 and 1, predicting whether the transaction is fraudulent (closer to 1) or normal (closer to 0).

3.4.6 XGBoost Module

A gradient boosting-based machine learning model is utilized to detect fraudulent transactions by learning complex decision boundaries from tabular transaction data. XGBoost constructs an ensemble of decision trees that iteratively improve the classification performance by minimizing the prediction error at each stage.

Module Architecture

1. **Ensemble of Gradient Boosted Trees:** This core component of the model consists of multiple decision trees built in sequence. Each new tree aims to correct the errors made by the previous ones by focusing on instances that were misclassified, thus improving the model's predictive performance.

2. **Regularization and Subsampling:** During training, the model incorporates regularization techniques and data/feature subsampling to reduce the risk of overfitting and ensure generalization across different folds of data.

3. **Output Layer:** The final output is a probability score between 0 and 1 indicating the likelihood of fraud. A classification threshold can be applied to determine whether a transaction is predicted as fraudulent or not.

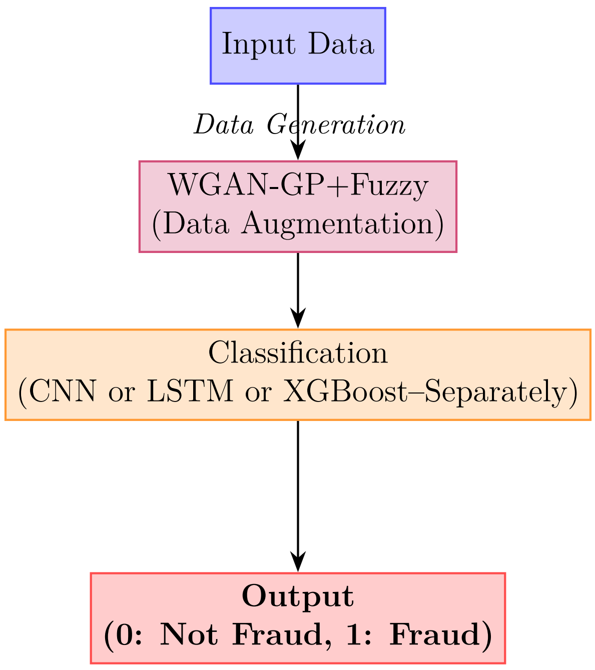


Figure 3.5: Hybrid Deep Fraud Network

In this study, we implement and evaluate several machine learning and deep learning architectures to address the problem of credit card fraud detection. The individual models considered include **Convolutional Neural Networks (CNN)**, **Long Short-Term Memory networks (LSTM)**, and **Extreme Gradient Boosting (XGBoost)**. Each of these models is trained independently on the balanced dataset to assess its individual classification performance.

The **CNN architecture** is designed to process reshaped tabular features as two-dimensional grids. It consists of convolutional layers followed by max-pooling and dense layers, ending with a sigmoid activation for binary classification. Similarly, the **LSTM model** is structured to handle sequential representations of the input data, capturing temporal dependencies that may reflect transaction patterns over time. The **XGBoost model** is employed as a powerful tree-based classifier optimized with regularization, known for its high performance on structured datasets.

To further improve detection accuracy, we propose a **hybrid architecture** that combines deep learning-based feature extraction with XGBoost classification. In this setup, either CNN or LSTM is used to extract high-level feature representations from the input data. These learned features are then concatenated with the original scaled tabular features and passed to an XGBoost classifier. This fusion strategy takes advantage of the representational capacity of deep networks and the decision-making strength of XGBoost, leading to enhanced robustness, generalization, and detection performance. Experimental results confirm that this hybrid approach yields superior accuracy compared to standalone models.

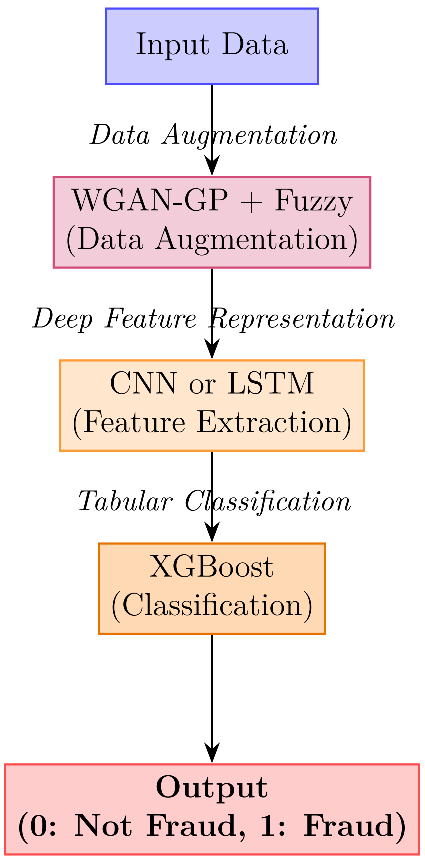


Figure 3.6: Hybrid architecture combining WGAN-GP, deep feature extraction, and XGBoost for fraud detection

Model	**Parameters**
CNN	- Number of convolutional layers: 1 - Filter size: (3×3), 32 filters - Pooling layers: MaxPooling (2×2) - Activation function: ReLU - Dropout rate: 0.5 - Fully connected layers: 2 - Dense units: [64, 1] - Final activation: Sigmoid - Learning rate: 0.001 - optimizer: adam
LSTM	- Number of LSTM layers: 2 - LSTM units per layer: [50, 100] - Activation function: tanh - Dropout rate: 0.5 - Fully connected layers: 1 - Final activation: Sigmoid - Learning rate: 0.001 - optimizer: adam
XGBoost	- Number of estimators: 200 - Learning rate: 0.05 - Maximum depth: 6 - Subsample: 0.8 - Colsample_bytree: 0.8 - Gamma: 1 - Regularization alpha (reg_alpha): 0.1 - Regularization lambda (reg_lambda): 1 - Scale_pos_weight: 1

Table 3.5: Models and their Parameters

3.5 Conclusion

By leveraging WGAN-GP for sophisticated data augmentation to address class imbalance, the proposed hybrid fuzzy-WGAN-GP based preprocessing step enriched the training data, enabling the models to better capture complex patterns, reduce overfitting, and improve sensitivity toward minority class detection. The experimental results validate the effectiveness of the proposed approach.The proposed model showed a significant improvement in performance metrics such as accuracy, recall, and detection rate. making it a promising and robust solution for fraud detection.

4.Experimental evaluation and discussion

4.1 Introduction

This concluding chapter represents a critical stage for validating the efficiency of the developed model through a series of rigorous experiments and systematic analyses. The chapter begins with a structured presentation of the development environment and experimental setup, including the characteristics of the dataset used, preprocessing steps, baseline models adopted for comparison, and the evaluation metrics applied. This is followed by a comprehensive discussion of the achieved results, supported by illustrative figures and precise quantitative data, highlighting the core value of each model component in enhancing its effectiveness and achieving optimal overall performance.

4.2 Experimental Setup

In this study, the TensorFlow library was employed as the primary framework for the development and implementation of deep learning models, owing to its robust and flexible toolset, as well as its advanced support for GPU-accelerated computation. These features facilitate the construction of complex model architectures and significantly expedite the training process.

All experiments and result analyses were conducted within the Google Colab environment, a free, cloud-based development platform that enables the execution of Python code directly through the browser, thereby eliminating the need for a complex local setup. To further accelerate training and benefit from extended computational resources, we utilized **Google Colab Pro**, which provides access to faster GPUs, longer session times, and increased memory, thus enhancing the efficiency and scalability of our experiments.

Given the variation in dataset sizes across experiments, a batch size ranging from 32 to 256 was adopted. The model was trained over 15 to 20 epochs, with the Early Stopping technique applied using a patience value of 5, in order to prevent overfitting and enhance the overall efficiency and generalization performance of the model

4.3 Data Pre-Processing

4.3.1 Feature Normalization

Before feeding the data into the neural network model, feature normalization is performed using `MinMaxScaler` from the `sklearn` library. This normalization transforms each feature into the range $[-1, 1]$ using the following equation:

$$X_{\text{scaled}} = 2 \cdot \left(\frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}} \right) - 1 \tag{4.1}$$

This transformation is essential because the GANs generator uses the `tanh` activation function in its output layer, which performs best when the inputs are scaled within the $[-1, 1]$ range.

4.4 Data Augmentation

4.4.1 Fréchet Inception Distance (FID)

FID is a statistical metric that quantifies the similarity between the distribution of real and synthetic credit card transaction features, typically modeled as multivariate Gaussians. It evaluates both the **mean** and the **covariance** structure of real versus generated data, providing a single score that reflects how well the GAN is capturing the real data distribution [15]. While FID is traditionally used to evaluate the quality of generated images in computer vision tasks, it is adapted in this context to measure the statistical similarity between real and synthetic transaction features, offering an interpretable and robust evaluation of GAN performance for fraud detection purposes.

Let:

- μ_r, Σ_r be the mean and covariance of the real transaction features.
- μ_g, Σ_g be the mean and covariance of the generated transaction features.

Then, the FID is computed as:

$$\text{FID} = \|\mu_r - \mu_g\|^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2\left(\Sigma_r \Sigma_g\right)^{1/2}\right) \tag{4.2}$$

A **low FID score** indicates high-quality synthetic data, which is useful for augmenting datasets and improving fraud detection models.

A **high FID score** suggests that the GAN is failing to capture the complexity of real fraud behavior.

European Dataset Analysis

The results on the European dataset clearly demonstrate that incorporating **Fuzzy Logic** significantly improves the quality of the generated data. The traditional **WGAN-GP** model recorded a relatively high FID score of 0.49, indicating weaker performance. Applying fuzzy logic solely to the discriminator (**Fuzzy + Discriminator**) led to a modest improvement, reducing the FID to 0.47.

The most notable enhancement was observed when fuzzy logic was applied exclusively to the generator (**Fuzzy + Generator**), with the FID dropping substantially to 0.08, reflecting the highest quality of generated data.

When fuzzy logic was integrated into both the generator and the discriminator (**Fuzzy + Generator + Discriminator**), a balanced performance was achieved, with an FID of 0.09.

Synthetic Dataset Analysis

For the synthetic dataset, the traditional GAN-GP model performed relatively poorly, yielding an FID of 0.60. Incorporating fuzzy logic into the discriminator only (**Fuzzy + Discriminator**) significantly improved the results, lowering the FID to 0.44.

The best FID score, 0.26, was obtained when fuzzy logic was applied to the generator only (**Fuzzy + Generator**), indicating a high quality of data generation.

The combined model with fuzzy logic in both the generator and discriminator (**Fuzzy + Generator + Discriminator**) showed good overall balance with an FID of 0.41.

Mobile Money Dataset Analysis

This dataset achieved the best overall results among the three. Even the baseline GAN-GP model produced strong outcomes with an FID of 0.20. Adding fuzzy logic to the discriminator (**Fuzzy + Discriminator**) further improved the data quality, reducing the FID to 0.12.

The best result was achieved when fuzzy logic was applied exclusively to the generator (**Fuzzy + Generator**), yielding an exceptional FID of 0.02.

Finally, the combined fuzzy logic model applied to both the generator and discriminator (**Fuzzy + Generator + Discriminator**) delivered a well-balanced performance with an FID of 0.04, making this configuration highly effective for complex financial data such as mobile money transactions.

Table 4.1: Experimental results with different methods and parameters across datasets

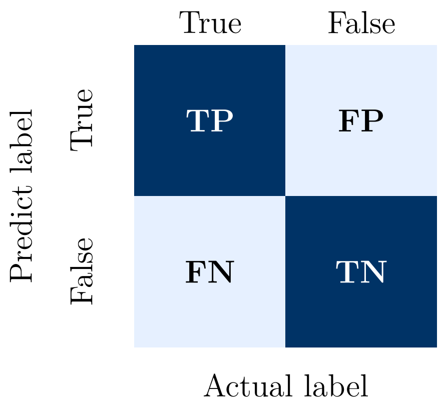
Dataset	FID	Latent dim	Learning rate	Batch size	n-critic	Fuzz dim	Method of data augmentation
European dataset	0.49	16	0.00089	128	3	8	GAN-GP
	0.47	16	0.00091	128	7	8	Fuzzy + Discriminator
	0.08	26	0.00087	128	7	8	Fuzzy + Generator
	0.09	16	0.00082	128	8	8	Fuzzy + Generator + Discriminator
Synthetic dataset	0.60	81	0.00090	32	9	8	GAN-GP
	0.44	53	0.00089	128	8	8	Fuzzy + Discriminator
	0.26	31	0.00084	128	6	8	Fuzzy + Generator
	0.41	44	0.00065	32	10	8	Fuzzy + Generator + Discriminator
Mobile money dataset	0.20	88	0.00099	128	6	8	GAN-GP
	0.12	49	0.00099	128	3	8	Fuzzy + Discriminator
	0.02	55	0.00026	128	8	8	Fuzzy + Generator
	0.04	47	0.00021	64	10	8	Fuzzy + Generator + Discriminator

4.5 Evaluation Metrics

4.5.1 Confusion Matrix

A **Confusion Matrix** is a tool used to evaluate the performance of a classification model. This matrix illustrates how the model classifies samples by comparing the true classes with the classes predicted by the model.

It is essentially a table that shows the actual counts of cases belonging to each class (such as positive and negative) and how they were classified (correctly or incorrectly).



- **True Positive (TP):** Normal transactions correctly predicted as normal.
- **False Positive (FP):** Fraudulent transactions incorrectly predicted as normal.
- **True Negative (TN):** Fraudulent transactions correctly predicted as fraud.
- **False Negative (FN):** Normal transactions incorrectly predicted as fraud.

4.5.2 Precision

Precision is a metric used to evaluate the performance of a classification model. It is defined as the ratio of correctly predicted positive instances (True Positives) to the total number of instances predicted as positive by the model (True Positives + False Positives):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.3}$$

Where:

- **TP (True Positives)**: The number of positive samples that were correctly classified as positive.
- **FP (False Positives)**: The number of negative samples that were incorrectly classified as positive.

4.5.3 Recall

Recall, also known as *True Positive Rate* or *Sensitivity*, is a metric used to evaluate a classification model's ability to identify all true positive cases.

Recall is defined as the ratio of the number of correctly classified positive samples (True Positives) to the total number of actual positive samples (True Positives + False Negatives):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.4}$$

4.5.4 F1-Score

The **F1-Score** is a metric used to evaluate the performance of classification models. It is defined as the *harmonic mean* between **Precision** and **Recall**.

This metric is used when it is necessary to achieve a balance between Precision and Recall, especially in cases of imbalanced classes, where the F1-Score can provide a fairer evaluation compared to individual metrics.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4.5}$$

4.5.5 Binary Cross-Entropy

Binary Cross-Entropy, also known as **Log Loss**, is a loss function that measures the difference between the true distribution of the classes and the distribution predicted by the model. It is derived from the concept of entropy in information theory and is used when the model outputs are probabilities ranging between 0 and 1.

Let us consider:

- $y \in \{0, 1\}$: the ground truth label.
- $\hat{y} \in (0, 1)$: the predicted probability that the sample belongs to the positive class.

Then, the Binary Cross-Entropy (BCE) loss is calculated as:

$$\text{BCE}(y, \hat{y}) = -\left[y \cdot \log(\hat{y}) + (1 - y) \cdot \log(1 - \hat{y})\right]$$

This loss function is widely used in binary classification tasks. It penalizes incorrect predictions more heavily when the model is confident but wrong, encouraging the model to improve its probabilistic predictions over time.

4.6 Experimental Results

4.6.1 European Dataset

The table presents the performance of various models on the European dataset using four methodologies: WGAN-GP, Fuzzy+Discriminator, Fuzzy+Generator, and Fuzzy+Both, with evaluation based on Precision, Recall, and F1-Score.

Under the GAN-GP methodology, the LSTM+XGB model achieved the best performance with an F1-Score of 0.9998, outperforming the other models.

In the Fuzzy+Discriminator approach, both XGB and LSTM+XGB stood out with F1-Scores equal to or above 0.9997, highlighting the effectiveness of combining fuzzy logic with the discriminator.

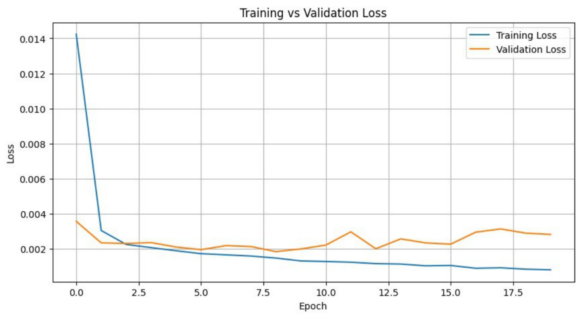
For Fuzzy+Generator, LSTM+XGB and CNN+XGB delivered the highest results with an F1-Score of 0.9996, showing that the fuzzy generator improves classification capability.

In the final methodology, which combines both Discriminator and Generator, performance was similar across models, with CNN+XGB and LSTM+XGB achieving stable results (F1-Score = 0.9996).

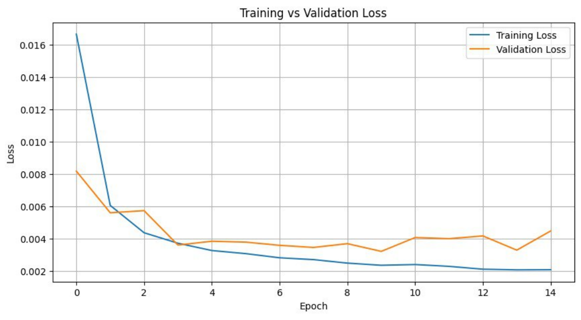
Overall, LSTM+XGB proved to be the most consistent and high-performing model across all methodologies, indicating that integrating deep learning (LSTM) with boosting models (XGB) offers strong results in complex classification tasks.

Table 4.2: Performance comparison of Different Models on the European Dataset

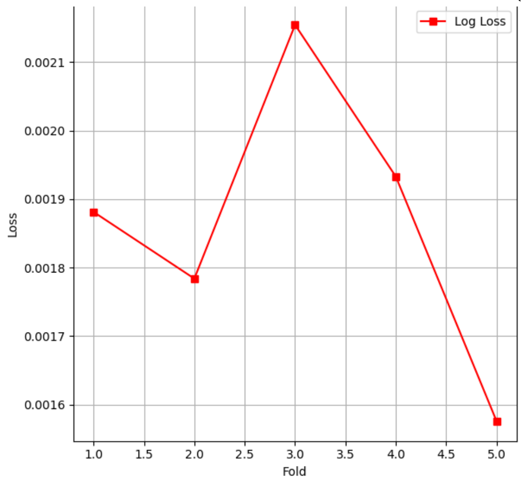
European Dataset				
Module	**Precision**	**Recall**	**F1-Score**	**Method of data augmentation**
LSTM	0.9997	0.9997	0.9997	WGAN-GP
CNN	0.9998	0.9984	0.9991	
XGB	0.9996	0.9996	0.9996	
CNN+XGB	0.9997	0.9997	0.9997	
LSTM+XGB	0.9998	0.9997	0.9998	
LSTM	0.9998	0.9996	0.9997	Fuzzy+Discriminator
CNN	0.9997	0.9970	0.9984	
XGB	0.9996	0.9996	0.9996	
CNN+XGB	0.9997	0.9997	0.9997	
LSTM+XGB	0.9998	0.9997	0.9998	
LSTM	0.9998	0.9991	0.9995	Fuzzy+Generator
CNN	0.9995	0.9985	0.9990	
XGB	0.9996	0.9995	0.9984	
CNN+XGB	0.9997	0.9995	0.9996	
LSTM+XGB	0.9998	0.9996	0.9996	
LSTM	0.9997	0.9996	0.9997	Fuzzy+Discriminator + Generator
CNN	0.9996	0.9986	0.9988	
XGB	0.9995	0.9994	0.9994	
CNN+XGB	0.9997	0.9996	0.9996	
LSTM+XGB	0.9997	0.9996	0.9996	



(a) LSTM

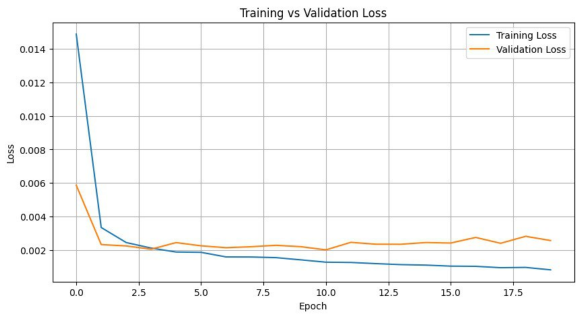


(b) CNN

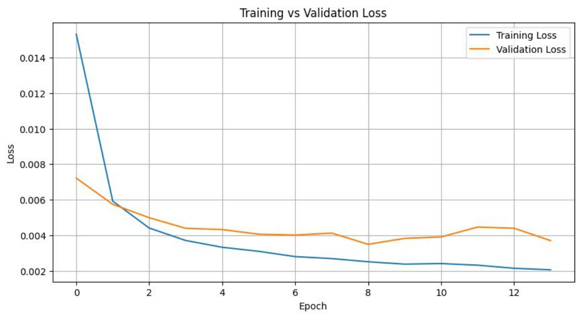


(c) XGBoost

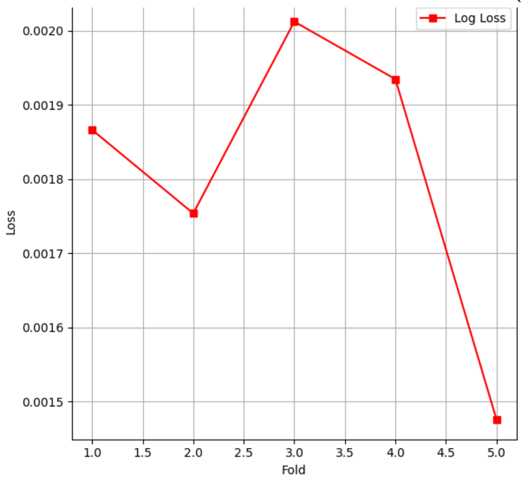
Figure 4.1: Method of WGAN-GP.



(a) LSTM

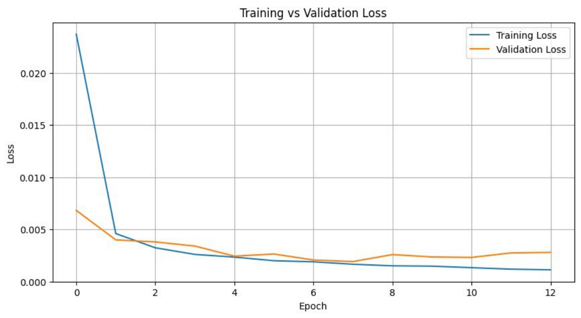


(b) CNN

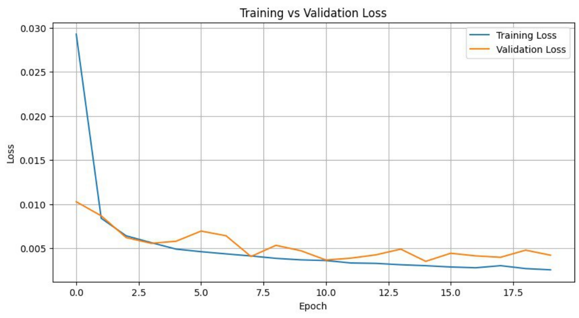


(c) XGBoost

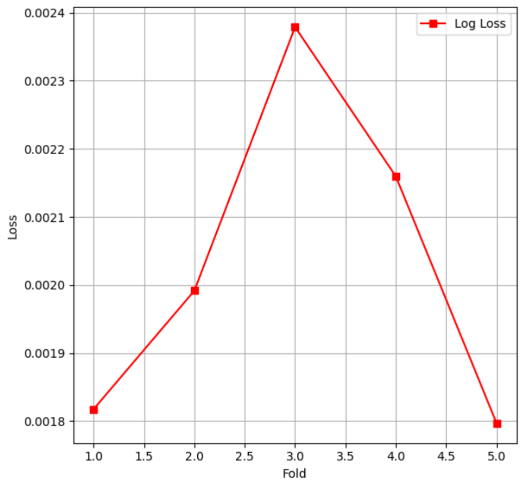
Figure 4.2: Method of Fuzzy and Discriminator.



(a) LSTM

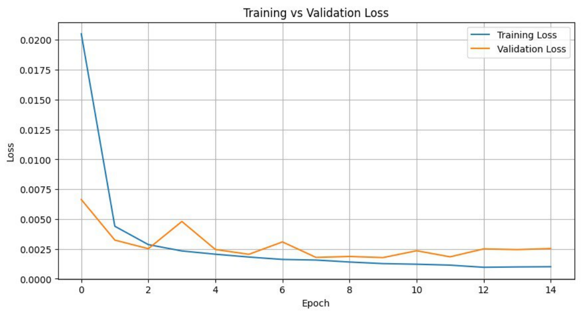


(b) CNN

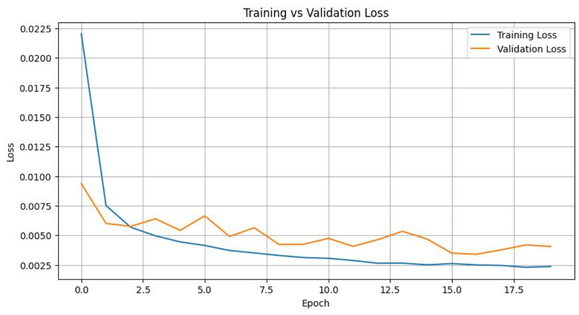


(c) XGBoost

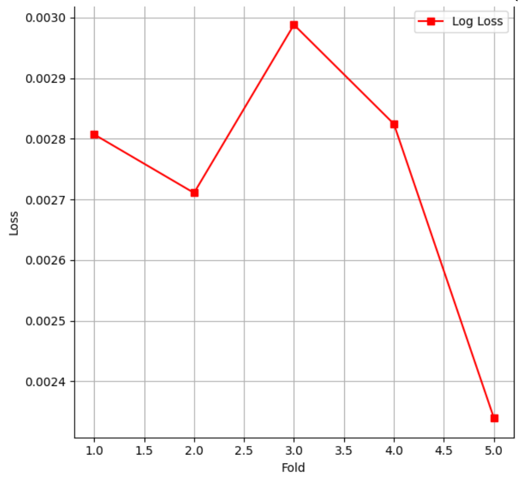
Figure 4.3: Method of Fuzzy and Generator.



(a) LSTM



(b) CNN



(c) XGBoost

Figure 4.4: Method of Fuzzy Generator and Discriminator.

4.6.2 Discussion 1 :

Across all four methods, consistent trends emerged:

- **LSTM and CNN models** exhibited rapid initial learning, with both training and validation losses decreasing significantly within the first few epochs. Training loss continued to decline steadily, while validation loss typically stabilized at a low value, indicating good generalization. Minor fluctuations were observed, but no significant overfitting was evident.

- **XGBoost**, assessed via *log loss per fold* during cross-validation, consistently achieved low log loss values, underscoring its strength in probabilistic classification. While fold-to-fold variability was present, it remained within acceptable bounds.

Comparative Performance of Data Preparation Methods

1. **Figure 4.1 – WGAN-GP:** This method consistently yielded the best overall performance. LSTM and CNN achieved low, stable validation losses (LSTM: 0.002–0.003; CNN: 0.003–0.0045), and XGBoost recorded the lowest log loss range (0.00048–0.00095). This reflects excellent model calibration and generalization.

2. **Figure 4.2 – Fuzzy and Discriminator:** This method produced the lowest validation loss for LSTM (0.002–0.0025) and competitive results for CNN (0.0035–0.0045). XGBoost log loss (0.0012–0.00215) was slightly higher than with W-GAN-GP but still remarkably low, making this method highly effective, especially for deep learning models.

3. **Figure 4.3 – Fuzzy and Generator:** While validation loss remained acceptable, both LSTM (0.003–0.004) and CNN (0.004–0.005) showed marginally reduced generalization. XGBoost performance declined slightly, with log loss ranging from 0.0018 to 0.0024. This method is functional but less optimal compared to the prior two.

4. **Figure 4.4 – Fuzzy Generator and Discriminator:** This hybrid method showed the least favorable performance. LSTM and CNN recorded higher validation losses (LSTM: 0.0025–0.0035; CNN: 0.004–0.005), and XGBoost yielded the highest log loss range (0.0023–0.0030), indicating suboptimal model calibration.

In essence, the **Fuzzy + Discriminator** method, particularly when leveraging the synergy of **LSTM** and **XGBoost**, emerges as the optimal approach for this **European dataset**, demonstrating superior classification results as evidenced by the confusion matrix, and strong generalization capabilities in the individual model loss curves.

Table 4.3: Results Confusion Matrix of Different Models

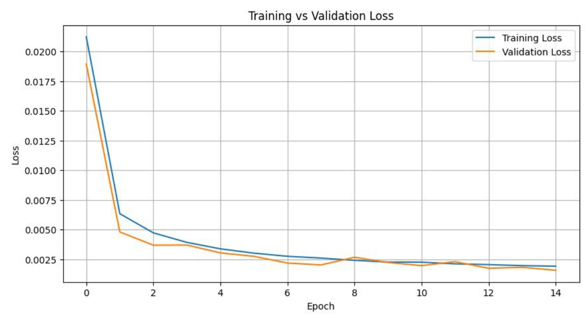
European Dataset					
Module	**TP**	**TN**	**FP**	**FN**	**Method of data augmentation**
LSTM	56844	56846	17	19	WGAN-GP
CNN	56849	56772	91	14	
XGB	56842	56841	21	20	
CNN+XGB	56851	56849	14	12	
LSTM+XGB	56845	56844	19	18	
LSTM	56851	56839	24	12	Fuzzy + Discriminator
CNN	56851	56853	110	13	
XGB	56844	56841	21	18	
CNN+XGB	56845	56848	15	18	
LSTM+XGB	56854	56848	15	9	
LSTM	56854	56810	53	9	Fuzzy + Generator
CNN	56834	56775	88	29	
XGB	56844	56838	24	18	
CNN+XGB	56846	56837	26	17	
LSTM+XGB	56849	56843	20	14	
LSTM	56845	56843	20	18	Fuzzy + Generator+ Discriminator
CNN	56845	56756	107	20	
XGB	56834	56831	31	28	
CNN+XGB	56844	56840	23	19	
LSTM+XGB	56838	56840	23	25	

4.6.3 PaySim Synthetic Mobile-Money Transactions

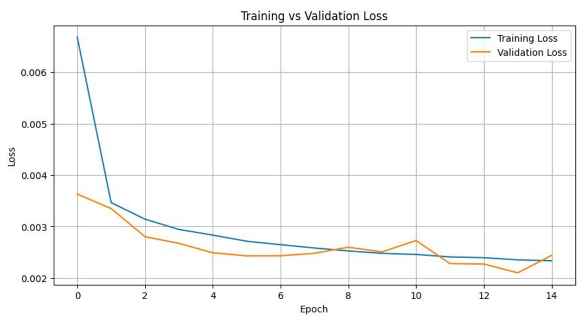
The table presents the performance of several classification models (LSTM, CNN, XGB, as well as combined models like CNN+XGB and LSTM+XGB) using precision, recall, and F1-score metrics under four different settings of the Fuzzy-WGAN-GP technique. The results show that all models achieve excellent performance with very high values exceeding 0.99 across all metrics, indicating a strong ability to distinguish between classes. When using the traditional WGAN-GP, the XGB model and the combined models stand out with the highest recall values reaching 0.9997, reflecting good accuracy in detecting positive cases. Applying fuzzy logic to the discriminator (Fuzzy + Discriminator) or the generator (Fuzzy + Generator) does not result in significant performance differences, with the XGB-supported models remaining stable and outstanding. When fuzzy logic is applied to both the generator and the discriminator (Fuzzy + Gen + Disc), a slight improvement in the recall of the LSTM model is observed, while the performance of the XGB models and combined models remains at the highest levels. Overall, the results indicate that using Fuzzy-WGAN-GP with its various settings maintains very strong performance with slight improvements in some models, highlighting the strength of combining traditional models with artificial generation and fuzzy logic techniques to enhance detection accuracy.

Table 4.4: Performance comparison of Different Models on the PaySim Synthetic dataset

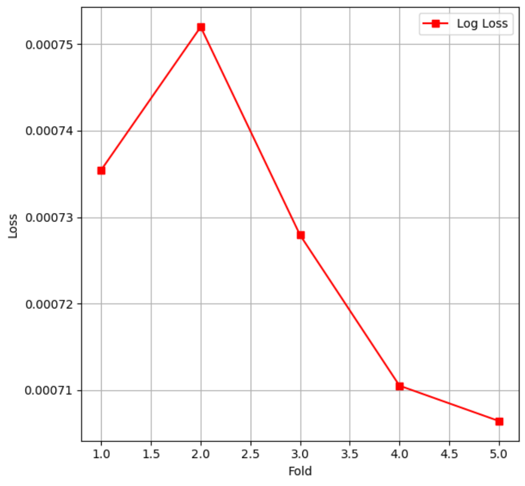
PaySim Synthetic Mobile-Money Transactions				
Model	**Precision**	**Recall**	**F1-Score**	**Method of data augmentation**
LSTM	0.9999	0.9991	0.9995	WGAN-GP
CNN	0.9999	0.9994	0.9997	
XGB	0.9999	0.9996	0.9997	
CNN+XGB	0.9999	0.9997	0.9998	
LSTM+XGB	0.9999	0.9997	0.9998	
LSTM	0.9999	0.9993	0.9996	Fuzzy+ Discriminator
CNN	0.9999	0.9974	0.9987	
XGB	0.9999	0.9996	0.9997	
CNN+XGB	0.9999	0.9997	0.9998	
LSTM+XGB	0.9999	0.9997	0.9998	
LSTM	0.9996	0.9996	0.9997	Fuzzy + Generator
CNN	0.9990	0.9990	0.9995	
XGB	0.9999	0.9996	0.9997	
CNN+XGB	0.9999	0.9997	0.9998	
LSTM+XGB	0.9999	0.9997	0.9998	
LSTM	0.9999	0.9996	0.9998	Fuzzy + Generator + Discriminator
CNN	0.9992	0.9994	0.9993	
XGB	0.9999	0.9996	0.9997	
CNN+XGB	0.9999	0.9997	0.9998	
LSTM+XGB	0.9999	0.9997	0.9998	



(a) LSTM

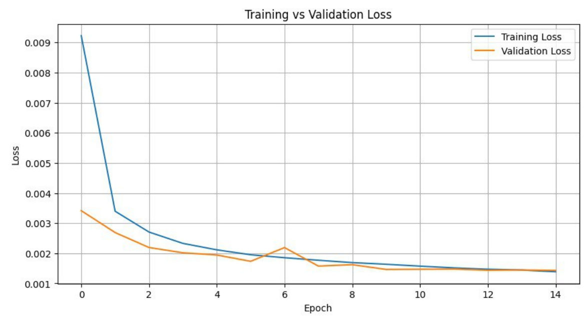


(b) CNN

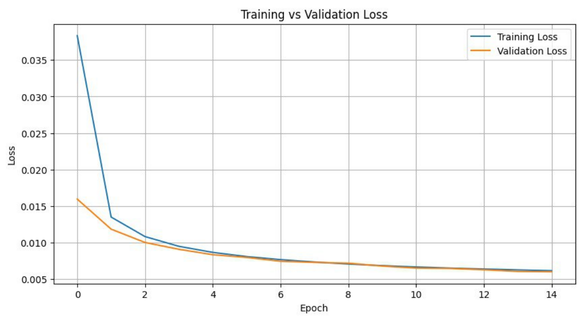


(c) XGBoost

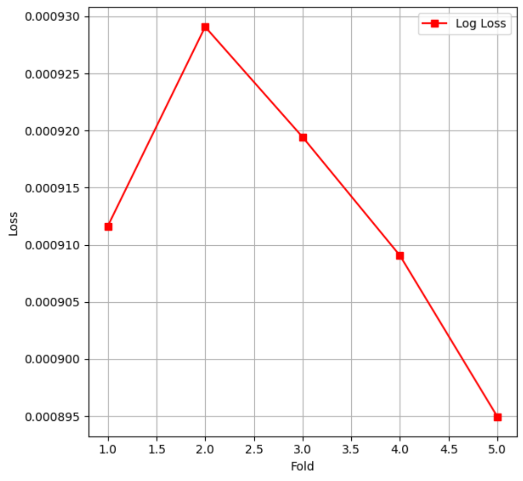
Figure 4.5: Method of WGAN-GP.



(a) LSTM

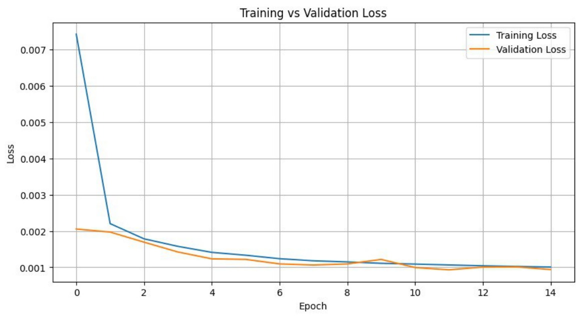


(b) CNN

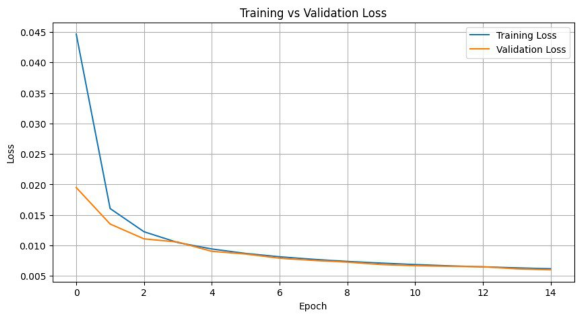


(c) XGBoost

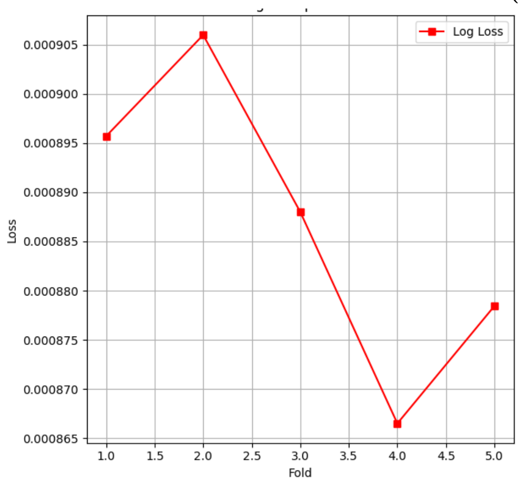
Figure 4.6: Method of Fuzzy and Discriminator.



(a) LSTM

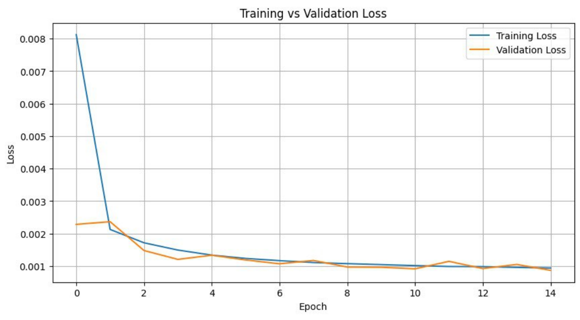


(b) CNN

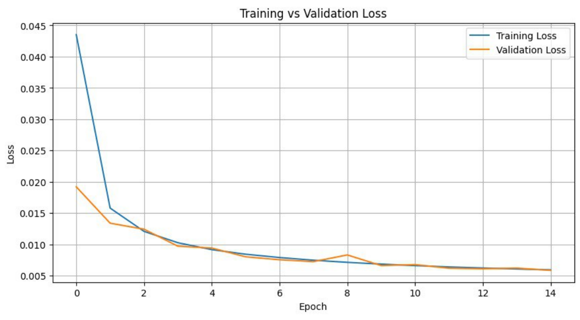


(c) XGBoost

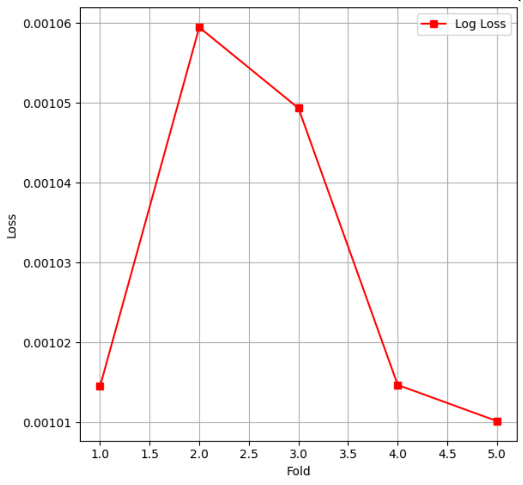
Figure 4.7: Method of Fuzzy and Generator.



(a) LSTM



(b) CNN



(c) XGBoost

Figure 4.8: Method of Fuzzy Generator and Discriminator.

4.6.4 Discussion 2 :

Across all four data preparation methods, consistent trends were observed:

- **LSTM and CNN** demonstrated a rapid initial learning phase. Both training and validation losses dropped significantly in the first few epochs. While training loss continued to decline steadily, validation loss stabilized at low levels, reflecting strong generalization capabilities. No major overfitting patterns were identified.
- **XGBoost**, evaluated using log loss per fold in cross-validation, consistently showed low log loss values. Although some variability existed across folds, it remained within acceptable limits, highlighting the model's robustness in probabilistic classification tasks.

Comparative Performance of Data Preparation Methods

1. **Figure 4.5 – WGAN-GP:** This method consistently yielded the best overall performance. LSTM validation loss stabilized in the range of (0.0025–0.003), while CNN showed similarly low validation loss between (0.002–0.003). XGBoost achieved exceptionally low log loss values ranging between (0.00070-0.000775), reflecting outstanding probabilistic calibration. Overall, GAN-GP proved highly effective across all models, especially for XGBoost.
2. **Figure 4.6 – Fuzzy and Discriminator:** This approach resulted in the lowest validation loss for LSTM, stabilizing between (0.0015–0.002), indicating excellent generalization. CNN also performed strongly with a validation loss around (0.002–0.003). XGBoost log loss ranged btween (0.00089–0.00093), slightly higher than GAN-GP but still demonstrating strong calibration. This method is particularly effective for LSTM while maintaining competitive results across the other models.
3. **Figure 4.7 – Fuzzy and Generator:** LSTM performance remained solid, with validation loss in the range of (0.0015–0.002). However, CNN exhibited a decline in generalization capability, with higher validation losses around (0.005–0.007). XGBoost log loss values were still low, between (0.000865–0.000905), though slightly inferior to the top-performing methods. Overall, while effective for LSTM and XGBoost, this method was comparatively weaker for CNN.
4. **Figure 4.8 – Fuzzy Generator and Discriminator:** This combined method demonstrated the least favorable performance. LSTM maintained low validation losses similar to previous methods, around (0.0015–0.002). However, CNN again showed higher validation loss values, stabilizing between (0.005–0.007). XGBoost recorded the highest log loss values of all methods, ranging from (0.00101–0.00106). These results suggest that this hybrid method was less effective in optimizing CNN and XGBoost performance, although LSTM results remained competitive.

While **WGAN-GP** excels in providing the lowest log loss for standalone *XGBoost*, and **Fuzzy + Generator** achieves the fewest false negatives with standalone *LSTM*, the **Fuzzy + Discriminator** method—particularly when leveraging the combined power of *LSTM* and *XGBoost*—emerges as the most effective overall strategy for robust and accurate classification on the **PaySim Synthetic Mobile-Money Transactions** dataset.

Table 4.5: Results Confusion Matrix of Different Models

PaySim Synthetic Mobile-Money Transactions					
Module	**TP**	**TN**	**FP**	**FN**	**Method data augmentation**
LSTM	1270867	1269692	1189	15	
CNN	1270865	1269565	1316	17	
XGB	1270857	1270428	453	24	WGAN-GP
CNN+XGB	1270844	1270496	385	38	
LSTM+XGB	1270844	1270520	361	38	
LSTM	1270868	1270038	843	14	
CNN	1270849	1267593	3288	33	
XGB	1270854	1270378	503	27	Fuzzy + Discriminator
CNN+XGB	1270842	1270490	391	27	
LSTM+XGB	1270844	1270542	339	38	
LSTM	1270877	1270232	649	5	
CNN	1270861	1267722	3159	21	
XGB	1270853	1270385	496	27	Fuzzy + Generator
CNN+XGB	1270836	1270506	375	46	
LSTM+XGB	1270854	1270512	369	28	
LSTM	1270864	1270316	565	18	
CNN	1270862	1267543	3338	20	
XGB	1270852	1270322	559	29	Fuzzy + Generator+ Discriminator
CNN+XGB	1270837	1270492	389	45	
LSTM+XGB	1270856	1270494	387	26	

4.6.5 The Synthetic Transactions Dataset

The table presents the performance of several models on the Synthetic dataset using four different methodologies: WGAN-GP, Fuzzy+Discriminator, Fuzzy+Generator, and Fuzzy+Both, with evaluation based on Precision, Recall, and F1-Score.

Under the WGAN-GP methodology, XGB, CNN+XGB, and LSTM+XGB achieved similarly high performance, all scoring an F1-Score of 0.9988. This indicates the effectiveness of hybrid models in handling generated data.

In the Fuzzy+Discriminator approach, the LSTM+XGB model continued to perform exceptionally well (F1-Score = 0.9988), along with XGB and CNN+XGB, suggesting that the integration of a fuzzy discriminator maintains high accuracy without introducing significant performance gains.

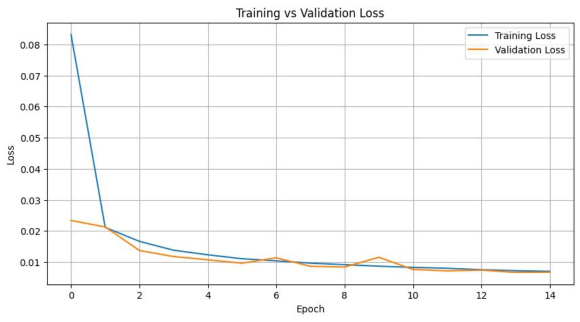
With Fuzzy+Generator, LSTM+XGB maintained top performance (F1-Score = 0.9988), followed closely by XGB and CNN+XGB, highlighting the role of the fuzzy generator in enhancing results.

Finally, in the methodology combining both Discriminator and Generator, results were very close, with XGB, CNN+XGB, and LSTM+XGB each achieving an F1-Score of 0.9987, while standalone LSTM and CNN models showed a slight decrease in performance.

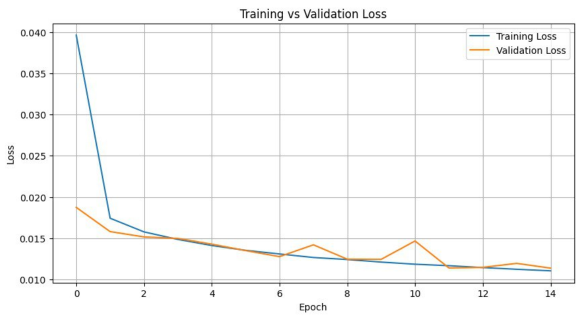
Overall, LSTM+XGB remains the top-performing and most stable model across all methodologies, demonstrating its efficiency in handling synthetic data, especially when integrated with techniques like WGAN-GP and fuzzy logic.

Table 4.6: Performance comparison of Different Models on the Synthetic Transactions Dataset

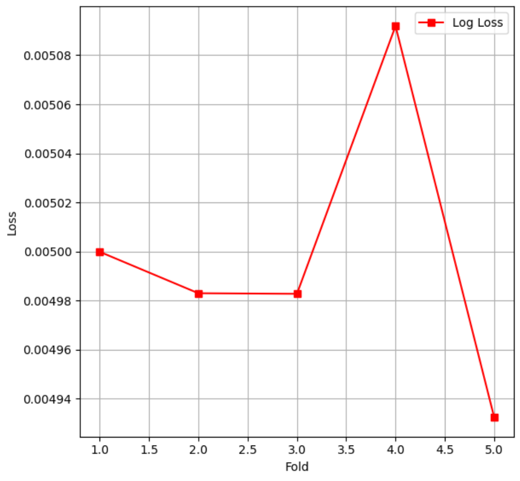
Synthetic Transactions Dataset				
Module	**Precision**	**Recall**	**F1-Score**	**Method of data augmentation**
LSTM	0.9997	0.9965	0.9981	WGAN-GP
CNN	0.9997	0.9933	0.9967	
XGB	0.9994	0.9982	0.9988	
CNN+XGB	0.9995	0.9982	0.9988	
LSTM+XGB	0.9994	0.9982	0.9988	
LSTM	0.9998	0.9965	0.9982	Fuzzy + Discriminator
CNN	0.9996	0.9937	0.9966	
XGB	0.9994	0.9982	0.9988	
CNN+XGB	0.9993	0.9980	0.9987	
LSTM+XGB	0.9993	0.9982	0.9988	
LSTM	0.9998	0.9953	0.9975	Fuzzy + Generator
CNN	0.9996	0.9925	0.9957	
XGB	0.9994	0.9976	0.9986	
CNN+XGB	0.9994	0.9977	0.9986	
LSTM+XGB	0.9994	0.9981	0.9988	
LSTM	0.9992	0.9936	0.9964	Fuzzy + Discriminator + Generator
CNN	0.9994	0.9977	0.9986	
XGB	0.9994	0.9987	0.9987	
CNN+XGB	0.9997	0.9952	0.9987	
LSTM+XGB	0.9997	0.9952	0.9987	



(a) LSTM

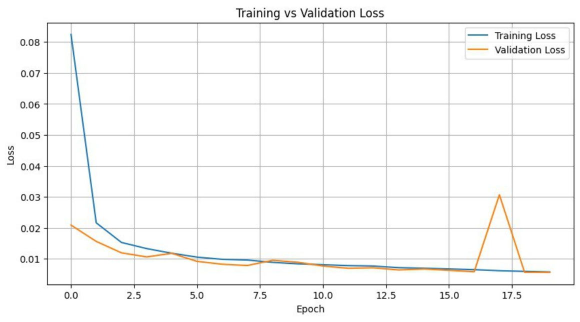


(b) CNN

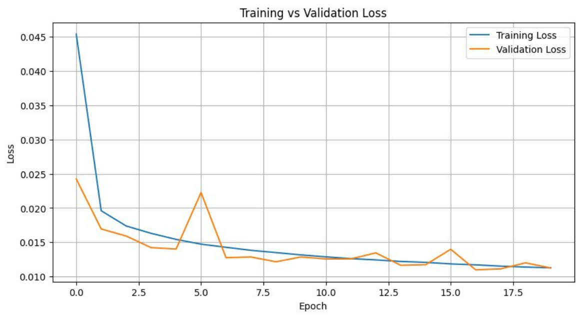


(c) XGBoost

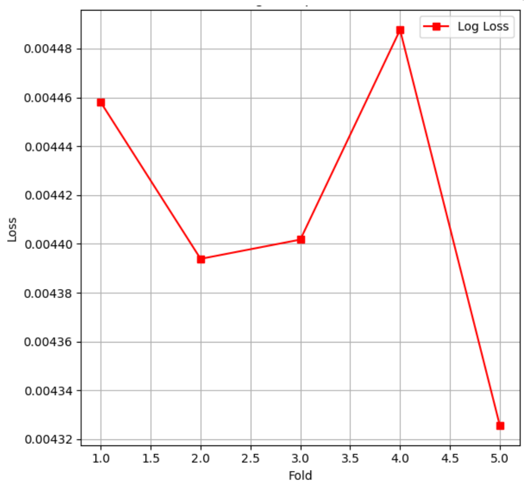
Figure 4.9: Method of WGAN-GP.



(a) LSTM

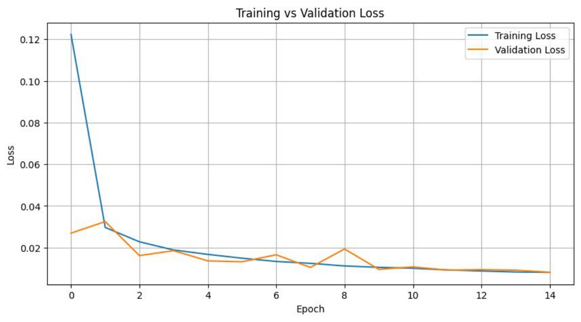


(b) CNN

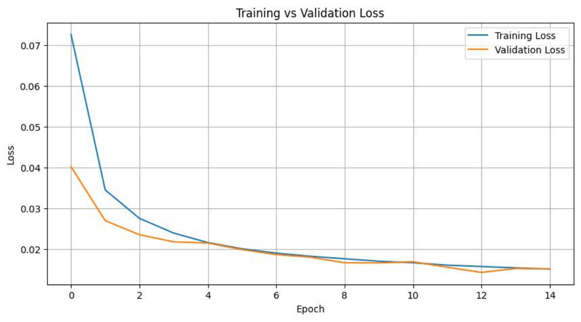


(c) XGBoost

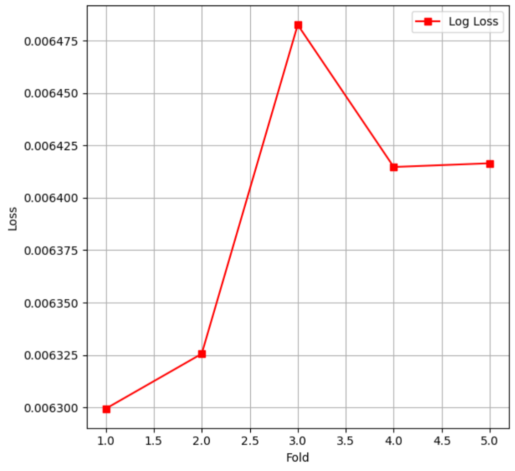
Figure 4.10: Method of Fuzzy and Discriminator.



(a) LSTM

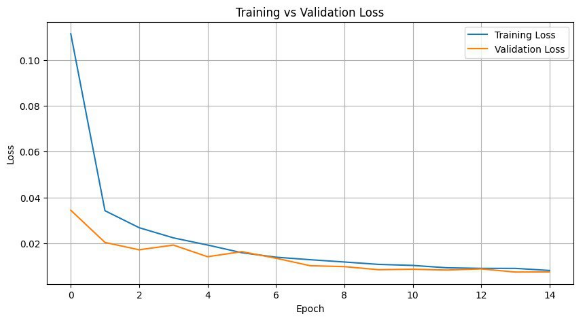


(b) CNN

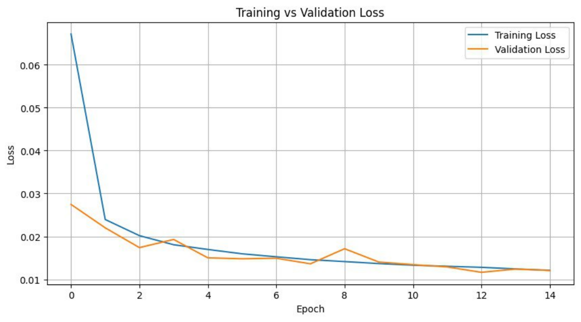


(c) XGBoost

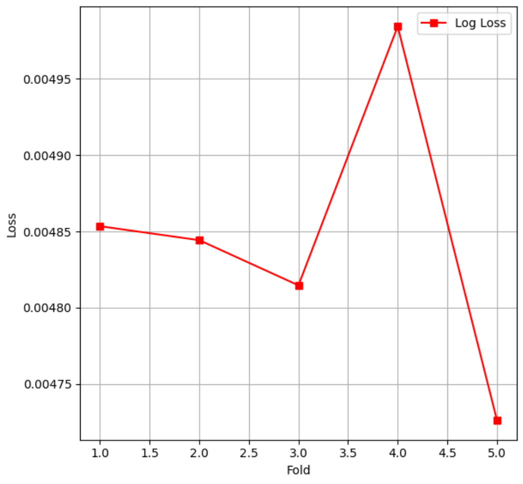
Figure 4.11: Method of Fuzzy and Generator.



(a) LSTM



(b) CNN



(c) XGBoost

Figure 4.12: Method of Fuzzy Generator and Discriminator

4.6.6 Discussion 3 :

Across all four data preparation methods, consistent trends were observed:

- **LSTM and CNN** models exhibited rapid initial learning, with both training and validation losses decreasing substantially within the first few epochs. Training loss continued to decline steadily, while validation loss typically stabilized at a low value, suggesting strong generalization. Minor fluctuations were observed, but no significant overfitting occurred.

- **XGBoost**, evaluated through log loss per fold in cross-validation, consistently achieved low log loss values, confirming its strength in probabilistic classification. While inter-fold variability was present, it remained within acceptable margins.

Comparative Performance of Data Preparation Methods

1. **Figure 4.9 – WGAN-GP:** This method demonstrated strong overall performance. LSTM and CNN achieved low, stable validation losses (LSTM: ~0.01; CNN: ~0.015), while XGBoost attained very low log loss values (0.0049–0.0050). This reflects solid model generalization and effective training across all architectures.

2. **Figure 4.10 – Fuzzy and Discriminator:** This method produced the lowest log loss for XGBoost (0.0043–0.0045) and competitive results for LSTM (~0.01) and CNN (~0.015). It stands out as a particularly strong approach, especially for boosting XGBoost performance.

3. **Figure 4.11 – Fuzzy and Generator:** While results remained acceptable, this variant showed slightly reduced performance. LSTM validation loss increased (to ~0.02), and XGBoost log loss rose to 0.0063–0.0064. CNN performance remained stable (~0.015), though generalization was marginally lower overall.

4. **Figure 4.12 – Fuzzy and Generator and Discriminator:** This hybrid method exhibited the least favorable outcomes. LSTM and CNN experienced higher validation losses (LSTM: ~0.02; CNN: ~0.015), and XGBoost log loss, while still low, reached 0.0048–0.0049 with some peaks above 0.0049. This suggests relatively weaker model calibration compared to other methods.

For the **Synthetic Transactions Dataset**, the **WGAN-GP** method with a combined **CNN+XGB** model stands out for its balanced and robust classification accuracy based on the confusion matrix. However, when considering individual model optimization, the *Fuzzy Discriminator* method provides the most precise probabilistic predictions for `XGBoost`.

Table 4.7: Results Confusion Matrix of Different Models

Synthetic Transactions Dataset					
Module	**TP**	**TN**	**FP**	**FN**	**Method of data augmentation**
LSTM	368449	367250	1299	100	WGAN-GP
CNN	368451	366248	2301	98	
XGB	368311	367800	748	237	
CNN+XGB	368348	367883	666	201	
LSTM+XGB	368321	367876	673	228	
LSTM	368490	367245	1304	59	Fuzzy + Discriminator
CNN	368377	367820	729	242	
XGB	368321	367861	687	227	
CNN+XGB	368403	366224	2325	146	
LSTM+XGB	368305	367880	669	244	
LSTM	368474	366818	1731	75	Fuzzy + Generator
CNN	368172	365802	2747	377	
XGB	368348	367498	1050	200	
CNN+XGB	368329	367714	835	220	
LSTM+XGB	368340	367860	689	209	
LSTM	368508	366831	1718	41	Fuzzy + Generator+ Discriminator
CNN	368322	366187	2362	277	
XGB	368335	367821	727	212	
CNN+XGB	368338	367793	756	211	
LSTM+XGB	368300	367870	679	249	

4.6.7 Conclusion

This experimental chapter has demonstrated the effectiveness of the proposed hybrid system that combines the WGAN-GP algorithm with Fuzzy Logic in enhancing credit card fraud detection. Through a series of rigorous and comparative experiments conducted on three different datasets, it was shown that integrating Fuzzy Logic into both the generator and the discriminator significantly improved the quality of the generated synthetic data by reducing the FID score and achieving a balanced data distribution .

What distinguishes this hybrid approach is its clear superiority over traditional methods (such as standard WGAN-GP), as the results were more stable and accurate. This was particularly evident in the strong performance of the hybrid LSTM+XGBoost model, which achieved the best scores in precision, recall, and F1-score, outperforming both standalone and conventional models.

Moreover, the proposed system demonstrated exceptional ability in handling imbalanced data—a common challenge in fraud detection applications—making it an effective and innovative alternative to previously adopted methods. This success highlights the importance of combining deep generative models with fuzzy logic to develop intelligent systems capable of addressing complex challenges in the field of financial security.

5.General Conclusion

This thesis tackled one of the most pressing challenges in the field of financial technology: credit card fraud detection, a domain where even small improvements can translate into significant financial savings and increased consumer confidence. Traditional machine learning methods have achieved reasonable success in detecting fraudulent transactions, but they often struggle with real-world conditions, especially the class imbalance problem, where fraudulent transactions account for less than 0.1 % of the total dataset. This results in biased models that prioritize accuracy on the majority class while failing to detect rare but critical fraud cases.

To address this, we proposed a hybrid deep learning system that brings together the strengths of Generative Adversarial Networks (GANs)—specifically Wersstein loss GAN with Gradient Penalty (WGAN-GP)—and Fuzzy Logic. The incorporation of Spectral Normalization into the GAN framework helped ensure more stable training by controlling the spectral norm of the discriminator's weight matrices, thus reducing training pathologies such as exploding or vanishing gradients.

Furthermore, Fuzzy Logic was integrated at different levels of the architecture (generator, discriminator, and both) to allow the model to better handle uncertainty and ambiguity, which are inherent in real-world financial data. This enabled the system to not only generate more realistic synthetic fraudulent transactions but also make more nuanced decisions during classification. We evaluated our approach on three diverse datasets: European Credit Card Dataset – A widely used benchmark in financial fraud detection. PaySim Synthetic Mobile Money Dataset – Simulated mobile transactions representing realistic financial behaviors. A Synthetic Dataset generated using Sparkov – To test the scalability and adaptability of our model. Our evaluation metrics included the F1-score, which is critical in imbalanced scenarios, and the Fréchet Inception Distance (FID), which assessed the quality and realism of the generated samples. We also employed standard classification metrics like Precision, Recall, and Confusion Matrix analysis across several deep learning models including CNN, LSTM, and XGBoost. The results demonstrated:
A significant improvement in F1-score when using augmented data generated by our hybrid model.

Better generalization performance across unseen data.
A noticeable reduction in false negatives, which is crucial in fraud detection systems.
Enhanced model stability and faster convergence during training due to the integration of spectral normalization.

By generating high-quality synthetic samples and refining the discrimination process through fuzzy reasoning, our system successfully addressed both data imbalance and model robustness challenges. The adaptability of the model to different datasets also proves its potential for real-world deployment in various financial institutions and transaction systems.

Limitations

Despite these promising results, there are some limitations to our approach:

1. *Code Complexity and Execution Time:* The proposed architecture requires significant computational resources and long execution times, particularly during the training of GAN-GP and deep learning models such as CNN and LSTM. This is mainly due to the complexity of the model structures and the need for tuning multiple hyperparameters.

2. *Hyperparameter Sensitivity:* Model performance is highly sensitive to the choice of hyperparameters, such as learning rate, number of layers, filter sizes, and number of units in LSTM. Finding the optimal settings requires a large number of experiments, which increases the computational burden.

3. *Slight fluctuations between training and validation:* Occasional slight fluctuations were observed between the training and validation performance. Although these do not affect the final model performance, they pose a minor challenge in accurately evaluating the model.

Future Work Applying the framework to other domains with similar data imbalance issues such as cybersecurity, medical diagnostics, and insurance fraud.

Final Remark In conclusion, this thesis contributes a novel and practical solution to the ongoing challenge of credit card fraud detection. By combining advanced generative models with intelligent reasoning techniques, we provide a robust and scalable framework that not only enhances the detection capabilities of AI systems but also aligns with the dynamic and complex nature of modern financial fraud.

Bibliography

[1] K. Afriyie et al. “A Supervised Machine Learning Algorithm for Detecting and Predicting Fraud in Credit Card Transactions.” In: *Decision Analytics Journal* 6 (2023), p. 100163. URL: https://www.sciencedirect.com/science/article/pii/S2772662223000036.

[2] F. K. Alarfaj et al. “Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms.” In: *IEEE Access* 10 (2022), pp. 1–1. DOI: 10.1109/ACCESS.2022.3166891. URL: https://www.researchgate.net/publication/359928222_Credit_Card_Fraud_Detection_Using_State-of-the-Art_Machine_Learning_and_Deep_Learning_Algorithms.

[3] N. S. Alfaiz and S. M. Fati. “Enhanced Credit Card Fraud Detection Model Using Machine Learning.” In: *Electronics* 11.4 (2022), p. 662. DOI: 10.3390/electronics11040662. URL: https://www.mdpi.com/2079-9292/11/4/662.

[4] Ibtissam Benchaji, Samira Douzi, and Bouabid El Ouahidi. “Credit Card Fraud Detection Model Based on LSTM Recurrent Neural Networks.” In: *Journal of Advances in Information Technology* 12.2 (2021), pp. 113–118. ISSN: 1798-2340. URL: https://www.academia.edu/73195427/Credit_Card_Fraud_Detection_Model_Based_on_LSTM_Recurrent_Neural_Networks.

[5] Siddhartha Bhattacharyya et al. “Data mining for credit card fraud: A comparative study.” In: *Decision Support Systems* 50.3 (2011), pp. 602–613. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167923610001326.

[6] Richard J. Bolton and David J. Hand. “Statistical Fraud Detection: A Review.” In: *Statistical Science* 17.3 (2002), pp. 235–255. DOI: 10.1214/ss/1042727940. URL: https://www.academia.edu/21686506/Statistical_Fraud_Detection_A_Review.

[7] Nitesh V. Chawla et al. “SMOTE: Synthetic Minority Over-sampling Technique.” In: *Journal of Artificial Intelligence Research* 16 (2002), pp. 321–357. URL: https://www.researchgate.net/publication/220543125_SMOTE_Synthetic_Minority_Over-sampling_Technique.

[8] Tianqi Chen and Carlos Guestrin. “XGBoost: A Scalable Tree Boosting System.” In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), pp. 785–794. DOI: 10.1145/2939672.2939785. URL: https://arxiv.org/abs/1603.02754.

[9] Levi Corallo and Aparna S. Varde. “Optical Character Recognition and Transcription of Berber Signs from Images in a Low-Resource Language Amazigh.” In: *arXiv preprint* arXiv:2303.13549 (2023). URL: https://arxiv.org/abs/2303.13549.

[10] Shawni Dutta and Samir Kumar Bandyopadhyay. "Detection of Fraud Transactions Using Recurrent Neural Network during COVID-19." In: *Journal of Advanced Research in Medical Science & Technology* 7.3 (2020), pp. 16–21. ISSN: 2394-6539. URL: `https://journals.indexcopernicus.com/api/file/viewByFileId/1160668`.

[11] William Paulo Ducca Fernandes et al. "Appellate Court Modifications Extraction for Portuguese." In: *Artificial Intelligence and Law* 28.3 (2020), pp. 327–360. URL: `https://www.researchgate.net/publication/334430718_Appellate_Court_Modifications_Extraction_for_Portuguese`.

[12] Ugo Fiore et al. "Using Generative Adversarial Networks for Improving Classification Effectiveness in Credit Card Fraud Detection." In: *Information Sciences* 479 (2019), pp. 448–455. URL: `https://www.sciencedirect.com/science/article/abs/pii/S0020025517311519`.

[13] J. H. Friedman. "Greedy Function Approximation: A Gradient Boosting Machine." In: *Annals of Statistics* 29 (2001), pp. 1189–1232. URL: `https://www.researchgate.net/publication/2424824_Greedy_Function_Approximation_A_Gradient_Boosting_Machine`.

[14] Ian Goodfellow et al. "Generative Adversarial Nets." In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. 2014, pp. 2672–2680. URL: `https://papers.nips.cc/paper_files/paper/2014/file/5ca3e9b122f61f8f06494c97b1afc`
`Paper.pdf`.

[15] Martin Heusel et al. "GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium." In: *Advances in Neural Information Processing Systems*. Vol. 30. 2017, pp. 6626–6637. URL: `https://www.researchgate.net/publication/317930102_GANs_Trained_by_a_Two_Time-Scale_Update_Rule_Converge_to_a_Local_Nash_Equilibrium`.

[16] E. Ileberi and Y. Sun. "A Hybrid Deep Learning Ensemble Model for Credit Card Fraud Detection." In: *IEEE Access* (2024). URL: `https://www.researchgate.net/publication/385960113_A_Hybrid_Deep_Learning_Ensemble_Model_for_Credit_Card_Fraud_Detection`.

[17] Shweta Indolia et al. "Conceptual Understanding of Convolutional Neural Network – A Deep Learning Approach." In: *Procedia Computer Science* 132 (2018), pp. 679–688. URL: `https://www.researchgate.net/publication/325657562_Conceptual_Understanding_of_Convolutional_Neural_Network-_A_Deep_Learning_Approach`.

[18] B. Kasasbeh, B. Aldabaybah, and H. Ahmad. "Multilayer Perceptron Artificial Neural Networks-Based Model for Credit Card Fraud Detection." In: *Indonesian Journal of Electrical Engineering and Computer Science* 26 (2022), pp. 362–373. URL: `https://www.researchgate.net/publication/359650741_Multilayer_perceptron_artificial_neural_networks-based_model_for_credit_card_fraud_detection`.

[19] Abdul Rehman Khalid et al. *Enhancing Credit Card Fraud Detection: An Ensemble Machine Learning Approach.* Tech. rep. Glasgow Caledonian University, 2022. URL: `https://researchonline.gcu.ac.uk/ws/portalfiles/portal/83214139/83184093.pdf`.

[20] M.J. Madhurya et al. "Exploratory Analysis of Credit Card Fraud Detection Using Machine Learning Techniques." In: *Global Transitions Proceedings* 3 (2022), pp. 31–37. URL: `https://www.sciencedirect.com/science/article/pii/S2666285X22000425`.

[21] P. Mangla. *Credit Card Fraud Detection Using Spectral Clustering*. Available online. 2024. URL: https://pyimagesearch.com/credit-card-fraud-detection-using-spectral-clustering.

[22] I.D. Mienye and N. Jere. “Deep Learning for Credit Card Fraud Detection: A Review of Algorithms, Challenges, and Solutions.” In: *IEEE Access* 12 (2024), pp. 96893–96910. URL: https://www.researchgate.net/publication/382187222_Deep_Learning_for_Credit_Card_Fraud_Detection_A_Review_of_Algorithms_Challenges_and_Solutions.

[23] I.D. Mienye and T.G. Swart. “A Hybrid Deep Learning Approach with Generative Adversarial Network for Credit Card Fraud Detection.” In: *Technologies* 12 (2024), p. 186. URL: https://www.mdpi.com/2227-7080/12/10/186.

[24] M.A. Mim, N. Majadi, and P. Mazumder. “A Soft Voting Ensemble Learning Approach for Credit Card Fraud Detection.” In: *Heliyon* 10.3 (2024). URL: https://www.researchgate.net/publication/377900487_A_soft_voting_ensemble_learning_approach_for_credit_card_fraud_detection.

[25] P. Mrozek, J. Panneerselvam, and O. Bagdasar. “Efficient Resampling for Fraud Detection During Anonymised Credit Card Transactions with Unbalanced Datasets.” In: (2020), pp. 426–433. URL: https://www.researchgate.net/publication/348091709_Efficient_Resampling_for_Fraud_Detection_During_Anonymised_Credit_Card_Transactions_with_Unbalanced_Datasets.

[26] Ryan Nguyen, Shubhendu Kumar Singh, and Rahul Rai. “Fuzzy Generative Adversarial Networks.” In: *Journal of Machine Learning Research* 1 (2021), pp. 1–48. URL: https://www.researchgate.net/publication/355698007_Fuzzy_Generative_Adversarial_Networks.

[27] A. Patel et al. “Credit card fraud detection using machine learning: A comprehensive review.” In: *Proceedings of the International Conference on Intelligent Data Science Technologies and Applications (IDSTA 2018)*. IEEE, 2018, pp. 71–76.

[28] Payments Dive. *Payments Fraud Losses Projected to Reach $34B Over the Next Decade, Nilson Says*. Accessed: April 20, 2025. 2025. URL: https://www.paymentsdive.com/news/payments-fraud-losses-prevention-nilson-outlook/737440/?utm_source=chatgpt.com.

[29] J. R. Quinlan. “Induction of decision trees.” In: *Machine Learning* 1 (1986), pp. 81–106. URL: https://link.springer.com/article/10.1007/BF00116251.

[30] 33rd Square. *Anomaly Detection in Credit Card Fraud*. Available online. 2025. URL: https://www.33rdsquare.com/anomaly-detection-in-credit-card-fraud.

[31] Hao Sun et al. “Uncertainty calibration and quantification of surrogate model for estimating the machining distortion of thin-walled parts.” In: *The International Journal of Advanced Manufacturing Technology* 120.5 (2022), pp. 1–23. URL: https://www.researchgate.net/publication/358311525_Uncertainty_calibration_and_quantification_of_surrogate_model_for_estimating_the_machining_distortion_of_thin-walled_parts.

[32] Mehwish Syeda, Khurram Shabbir, and Mohammad Waseem Ahmad. “Credit Card Fraud Detection Using Time Series ARIMA Model.” In: *arXiv preprint arXiv:2009.07578* (2020). URL: https://arxiv.org/pdf/2009.07578.

[33] Mesay Mulugeta Taye. “Theoretical Understanding of Convolutional Neural Network: Concepts, Architectures, Applications, Future Directions.” In: *Computation* 11.3 (2023), p. 52. URL: https://www.mdpi.com/2079-3197/11/3/52.

[34] Abhishek Verma. *Generative Adversarial Network*. Accessed: 2025-05-30. July 2019. URL: https://medium.com/xebia-engineering/generative-adversarial-network-2b063f587bae.