A Generative Pattern Extraction Algorithm Tackling Ultra-Imbalanced Data in Financial Fraud Detection Systems

Dr. Fatima M
Professor of Artificial Intelligence
United Arab Emirates University (UAEU), Al Ain, UAE
Research Interests: AI for Smart Cities, Autonomous Systems, Robotics

&

Prof. Al-Nuaimi
Senior Lecturer, Cybersecurity
University of Dubai, Dubai, UAE
Research Interests: Network Security, Cryptography, IoT Security

ABSTRACT

The process of financial fraud detection is increasingly becoming dependent on intelligent systems that are capable of identifying delicate, complex, and dynamic fraudulent activities in the large transactions environments. Fraudulent transactions tend to constitute less than 1 percent of the total volume in the real-world financial data. This renders the learning scenario very unequal. This difference is the challenge facing current research, with the traditional oversampling techniques, deep neural networks, and cost-sensitive learning approaches usually failing to retain the behavioral semantics of the fraudulent patterns. Such methods often lead to noisily or excessively simplified minority samples, which lead to poor generalization, large false-negative rates, and lower detection reliability in practice. To resolve these large issues, the current study proposes the Generative Pattern Extraction Algorithm (GPEA), a sequence-generative-ensemble network that is designed to enhance the representation and detection of minority fraud cases. The proposed system employs a Bi-LSTM encoder in order to learn latent motifs to explain how fraudulent behavior might vary with time. It then generates real minority-class samples in the learnt latent space by using a Conditional Generative Adversarial Network (CGAN). These more advanced representations are then added to a LightGBM -XGBoost ensemble classifier, which is used to distinguish more accurately between fraud and legitimate transactions. The significance of this technique is that it preserves the semantics of fraud, the learning in the minority class is easier, and it is also easier to detect fraud in highly asymmetric scenarios. The results reveal that GPEA consistently outperforms deep learning baselines, ensemble approaches, and the best imbalance management algorithms available. The proposed strategy decreases the false negative rate by 20% while maintaining the industry-leading fraud detection accuracy of 14%, the remarkable recall of 12-08%, the F1-score of 9-15%, and the overall accuracy of 14%. The model's overall detection rate of 96.3% shows that it can find fraud patterns that don't happen very often but are nonetheless very important. The experimental findings on benchmark financial fraud data indicate that our algorithm is significantly superior to oversampling baselines and deep-learning models both in the recall of the minorities, and in the precision/recall AUC, MCC, and the overall detection stability. It is also evident in the discussion that not only does GPEA help to categorize matters more accurately, but also develops patterns of fraud that are easy to understand. This renders it a handy aid to current systems of financial risk analysis and prevention of fraud.

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

Keywords: Financial Fraud Detection, Ultra-Imbalanced Data, Generative Pattern Extraction, Pattern-Aware Encoder, Bi-LSTM Encoder, Conditional GAN (CGAN), Minority-Class Augmentation, Ensemble Classifier, Synthetic Minority Generation, Fraud Behavior Semantics.

1. Introduction

The detection of fraud has become a pressing and unresolved problem of digital and financial ecosystems in the modern world owing to the immense multiplication of online transactions, electronic payment systems, and automated decision-making services. Since the volume of transactions has grown, the sophistication, adaptability as well as behavioral complexity of fraudulent activity has also grown, and in many instances, footprints left behind by the activity appear to resemble the normal user actions in order to go undetected. The marked feature of the type of fraud detection issues is the severe class imbalance of the real-world data, where the percentage of fraud cases among all records is very small. This bias greatly reduces the performance of the standard classification models that bias towards the majority class and cannot detect rare but high-impact cases of fraud. Moreover, fraud cases are also time-based and sequential in nature and are characterized by minute deviations in the timing and frequency of transactions along with behavioral patterns and not isolated anomaly. The current detection mechanisms are usually not able to capture these long-range relationships and changing behavioral patterns which leads to high false-negative rates and slow response. The issue of learning behavior-sensitive representations of fraud on a highly imbalanced sequence of data hence represents the core issue discussed in the paper.

Fraud detection methods A huge variety of such fraud detection methods have been suggested in the literature and include rule-based systems, machine-learning models, data-level resampling methods, deep-learning architectures [1], and generative methods. The first-generation fraud detection systems were quite dependent on rule-based systems that were specified by domain experts, manually constructed thresholds and heuristics to indicate suspicious transactions. Although these methods are computationally efficient and interpretable, they are not flexible and cannot be applied to new fraud trends. With the growth in data availability, supervised machine-learning models like logistic regression, support vector machines, decision trees, k-nearest neighbors and Random Forests gained widespread use because they can directly learn decision boundaries using historical data. Although they have been successful, these models are very sensitive to distorted classes and tend to give skewed predictions in case fraudulent cases are under sampled. To address this weakness, SMOTE, Borderline-SMOTE, ADASYN, and random oversampling [2], which are methods of oversampling, were designed to re-equilibrium the distribution of classes by producing artificial samples of the minority classes. In spite of the fact that such methods can enhance the recollection, they often distort features distributions, add noise, and do not preserve the actual behavioral peculiarities of fraud. More recently, deep-learning networks such as CNNs [3], LSTMs [4], Bi-LSTMs, and Transformer-based models have shown great ability in the ability to capture nonlinear interactions between features and time dependencies. Their efficacy is however limited in very skewed situations, where they lack enough minority-class samples to learn constant temporal and behavioral representations. Generative Adversarial Networks (GANs) [5][6] have also been considered as another option in data augmentation, yet much of the current literature input uses simple GAN models [7] on unstructured tabular data, resulting in unstructured and unnatural fraud samples because of the absence of pattern-conditioning and time-based context.

To address these constraints, this study presents three primary aims.

- The primary goal is to learn deep temporal and behavioral patterns that are related to fraudulent activity using a Bidirectional Long Short-Term Memory (Bi-LSTM) encoder [8]. The Bi-LSTM learns long-range dependencies and subtle sequence-level abnormalities that are predictive of fraud but ignored by other models by running transaction sequences forward and backward.
- The second task is to produce realistic and behavior-conserving samples of minority-classes with the help of a Conditional Generative Adversarial Network (CGAN) [9] trained in the latent space of the Bi-LSTM encoder. Producing samples in the latent space enables the model to be semantically consistent, still with its temporal and behavioral structure, without introducing the distortion of features that is traditionally added by standard oversampling models and raw-data GANs.
- The third goal is to improve the strength of classification and minimize false negatives by training a strong ensemble classifier based on LightGBM [10] and XGBoost [11][12][13] on a mixed real and CGAN-created samples dataset. This model combines the complementary advantages of the two gradient boosting models to have better discrimination performance on complex and imbalanced tabular data.

The suggested approach is combined into a single framework that is known as Generative Pattern Extraction Algorithm (GPEA), which unites representation learning, the process of generative modeling, and the process of ensemble classification. Contrary to the traditional methods that treat the class imbalance at either the data or the classifier-level individually, GPEA learns to represent temporal fraud representations, boosts minority-class learning in a structured latent space, and utilizes a powerful ensemble classifier to encourage the highest possible detection performance. The integrated design has the advantage of ensuring both that synthetic samples are statistically plausible and behaviorally meaningful, which enhances the extrapolation of synthetic samples to real-life fraud contexts. Through direct modeling of transaction sequences, as well as the use of latent-space generation, the suggested approach addresses the shortcomings of feature-level oversampling, and unconditioned generative models.

The rest of this paper will be structured in such a way that it provides the motivation, methodology, and assessment of the proposed approach in a systematic manner. Section II gives a thorough overview of current fraud detection methods, including classic rule-based, classical machine-learning, oversampling, deep-learning [14], and GAN-based data augmentation methods. Section III presents the proposed Generative Pattern Extraction Algorithm in detail along with the description of the architecture of the Bi-LSTM encoder, the latent-space Conditional GAN, and the LightGBM+XGBoost ensemble classifier. Section IV summarizes the experimental environment such as the datasets, preprocessing, model training policies, and evaluation measures used to measure the performance. In section V, the results of the experiment are presented and analyzed with an emphasis on the increase in detection accuracy, recall, and reduction in false-negatives, and the evaluation of the quality and realism of generated patterns of fraud. Lastly, section VI finishes the paper presenting a conclusion and summarizing the primary contributions and future research opportunities of generative modeling, sequential representation learning, and advanced fraud analytics.

2. Literature Survey

HaiChao Du et.al, [15] examines the AE-XGB-SMOTE-CGAN approach works far better than common machine learning models like K-Nearest Neighbors (KNN) and LightGBM. Experimental results indicate that it exceeds both baseline algorithms by around 2% in overall

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

accuracy (ACC), demonstrating enhanced classification reliability across both majority and minority classes. This improvement is quite important in situations where fraud detection is very uneven, because even minor increases in accuracy can lead to a big drop-in undetected fraudulent activity. When the decision threshold is set to 0.35, the Matthews Correlation Coefficient (MCC) for the AE-XGB-SMOTE-CGAN model goes up by an amazing 30% compared to KNN. MCC is one of the best ways to measure performance on imbalanced datasets, therefore the fact that the suggested hybrid algorithm learned relevant fraud-related patterns with such a big margin shows that it is better than other methods. This higher MCC score shows that there is a better balance between true positives, true negatives, false positives, and false negatives. The results show that the AE-XGB-SMOTE-CGAN method improves important performance measures like accuracy, true positive rate (TPR), true negative rate (TNR), and MCC, which means it can find more fraud. These improvements show that the system is better at finding fake transactions while still working well with real ones. So, the hybrid combination of autoencoders, gradient boosting, synthetic oversampling, and conditional generative modeling makes modern credit card fraud detection systems more reliable and stronger.

Masad A. Alrasheedi et.al, [16] study has identified various unique signs that differentiate fake tax records from valid ones, such as unusual deductions, differences in reported income, frequent patterns of transaction manipulation, and unusual filing practices. An encoder-based feature extraction technique effectively captures these fraud-related qualities by turning raw tax data into rich latent representations that show hidden abnormalities. To enhance minority-class learning, synthetic data augmentation techniques are utilized to produce supplementary samples that display infrequent fraudulent patterns. We tested several machine learning classifiers under two conditions: with synthetic augmentation and without it. This made it possible to fairly compare their performance when the data was not balanced. A Soft-Voting ensemble technique is used to make predictions more dependable by integrating the outputs of different classifiers to get a final judgment that is more stable and predictable. The experimental results show that the suggested Soft-Voting technique always beats solo classifiers. This is clear in the higher accuracy of fraud detection and the overall quality of the classification. The results show that using encoder-based pattern extraction with data augmentation and ensemble learning makes a model stronger, especially when trying to find complicated tax fraud.

Karthikeyan et.al, [17] method, called Chimp-Optimized Long Short-Term Memory Networks (ChOpt+LSTM), is done in two steps that happen one after the other. To begin with, an optimization strategy based on how chimpanzees act is used to find the most important traits for finding fraud. After that, these traits are used to teach a Long Short-Term Memory (LSTM) classifier model that was made just for finding cases of credit card fraud. A thorough comparative study revealed that the new ChOpt+LSTM method outperforms existing techniques in several significant performance metrics. It obtains a classification accuracy of 99.18%, a mean absolute error (MAE) of 25.7, and a mean squared error (MSE) of 16.3. It also gets scores of 98.54% for precision, 98.47% for recall, and 96.58% for F1. These results add to the evidence that using LSTM classifiers with dynamic optimization approaches is a good way to make bank fraud detection systems more accurate and reliable.

Chen, Y., Du et.al, [18] propose to predict financial fraud transactions using global enhanced graph convolution and Bidirectional LSTM, which we name GEGCN-BiLSTM. First, BiLSTM is used to find the temporal dependence in transactions that involve money laundering. It takes into account both the historical data and the data from the next time steps. Then, GEGCN is utilized to dig deeper into how distinct transactions are related in terms of their spatial global context. The output information from GEGCN will be used as the input for BiLSTM at each time stamp. This will combine time dependency with spatial dependence. The experimental results indicate that GEGCN-BiLSTM surpasses other comparative algorithms in both efficacy and importance, serving as a robust instrument for market transaction oversight.

Mohammed Tayebi, et.al, [19] research introduces an improved XGBoost algorithm for identifying fraudulent transactions by employing an intelligent method that optimizes the algorithm's hyperparameters via Bayesian optimization. We run a number of tests on two credit card datasets with both real and fake transactions to see how well our approach works. To avoid overfitting on datasets that aren't balanced, we used cross-validation, SMOTE, and random under-sampling. The highest result for Data 1 using SMOTE had an accuracy of 0.9996, a precision of 0.9406, a recall of 0.8740, an F-measure of 0.8740, and an AUC of 0.9879. The Random Under-sampling method worked best for Data 2, with an accuracy of 0.8325, a precision of 0.8294, a recall of 0.8378, an F-measure of 0.8336, and an AUC of 0.9088. These experimental findings show that our proposed method works better than competing machine learning models.

Ms. Hemangini Patel et.al, [20] research presents an improved framework derived from XGBoost to address critical challenges in fraud detection: class imbalance, the necessity for real-time processing, and the significance of model interpretability. We use the Kaggle Credit Card Fraud Dataset (which has 285,807 transactions and a fraud rate of 0.17%) and an SMOTE-ENN hybrid resampling method to make the class distributions equal and build features (from PCA on V1–V28 and temporal metrics) for effective training. The model uses cost-sensitive learning by giving false negatives 10 times more penalties, and it uses Youden's J statistic to make decision thresholds more precise (0.32). The results of the experiment reveal a 92% F1-score and an AUC-ROC of 0.98. These scores are better than those of LightGBM (89% F1) and LSTM networks (90% F1). The networks also have a significant inference latency of 12ms for real-time payment gateways. Analyzing mistakes shows ways to deal with false positives (by whitelisting high quantities) and false negatives (by using spend velocity features). A simulation of 1 million transactions per second on AWS shows that the framework can handle a throughput of 985,000 transactions per second, which proves that it can grow.

Lianhong Ding et.al, [21] study presents a LightGBM approach for detecting credit cards that is better since it uses an AutoEncoder. The approach takes the best parts of each component and combines them. It uses an AutoEncoder to reconstruct features in the dataset and the LightGBM algorithm to make the GBDT (Gradient Boosting Decision Tree) better at finding unusual data more quickly and accurately. In addition to the standard assessment measures, the F-measure, area under the curve (AUC), Matthew's correlation coefficient (MCC), and balanced classification rate (BCR) are also used. The suggested model was tested with two financial datasets to see how well it worked and how strong it was. The credit card fraud dataset with 31 characteristics shows that our model is far better than other models. It has a recall rate of 94.85%, which is 10.70% better than the top detection performance model, which only has a recall rate of 86%. Also, our model's BCR score is much better than those of other models.

Can Iscan et.al, [22] purpose of this study is to find fraud using the latest machine learning methods on data from Turkey's most popular e-wallet platform. After thoroughly examining the dataset's characteristics through feature engineering techniques, we discovered that the LightGBM method exhibited the highest detection accuracy of fraudulent activities, achieving 97% in the conducted testing. We also reached the important goal of cutting down on false alarms, which went from 13,024 to 6,249. This method led to the creation of a machine-learning model that tiny fraud detection teams can utilize.

HaiChao Du et al.[23] proposed a hybrid fraud detection system that combines autoencoders, XGBoost, SMOTE, and CGAN to solve the problem of a big class imbalance. The model makes fake fraud samples and improves feature representations, which raises MCC and accuracy. Still, there is a limit to how well things can be understood, and relying too much on explicit oversampling might lead to noise and duplication. GPEA, on the other hand, learns fraud-specific generating patterns directly in latent space, without aggressive oversampling. This makes recall more stable and generalizable across fraud behaviors that change over time.

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

Alrasheedi et al.[24] focused on finding tax fraud by using a soft-voting ensemble of classifiers, synthetic data augmentation, and encoder-based feature extraction. The method improves minority detection because the ensemble agrees, but it needs static feature encoders and bespoke augmentation. It also doesn't take into consideration how fraud changes over time. GPEA blends temporal motif learning with generative pattern extraction to enable for the adaptive detection of emerging fraud signatures. This is different from using static ensemble aggregation approaches.

ChOpt+LSTM, which was created by Karthikeyan et al.[25] is used to train an LSTM-based fraud classifier. It uses an optimization method based on how chimpanzees choose features. The model can get great recall and accuracy thanks to deep learning that is based on optimization. But other than choosing features, it's basically a mystery how to deal with ultra-imbalance. To make up for extreme imbalance, GPEA makes sure that performance and explainability are balanced by explicitly simulating minority fraud distributions using generative learning and interpretable temporal motifs.

3. Proposed Methodology

The Generative Pattern Extraction Algorithm (GPEA) framework attempts to solve the challenges of identifying ultra-imbalanced financial fraud by integrating an enhanced sequence-learning, generative modeling and ensemble classification framework into one framework. A Bi-LSTM encoder is utilized in the first stage in order to capture the temporal variation and the chain dependency that are present in transactional behavior. To distinguish between fraudulent and legitimate transactions, this encoder acquires long-range correlations, minute temporal anomalies, and repetitive behavioral patterns. Its bidirectional character allows it to view not only past but also future contextual information and extracts more profound and accurate latent patterns of fraud.

The encoded latent representations are fed to the Conditional GAN (CGAN) which produces a better synthetic sample of fraud that is conditioned on class labels and latent features. By steering clear of distortion of the overall structure of transaction information, this CGAN ensures the synthesized minority samples have behaviorally substantial frameworks. CGAN enhances the learning of the classifier in case of drastic imbalance conditions and assists the system in balancing the count of classes by generating natural but diverse patterns of frauds. LightGBM + XGBoost ensemble classifier which is a combination of gradient-boosted decision trees, allowing robust, stable yet high-precision fraud detection, is subsequently supplied with the enriched data which comprises both the authentic and fake samples. Whereas XGBoost is a powerful generalization and non-linear decision boundary, LightGBM is fast to train, interpret feature-importance, and high-dimension data. Their combination, be it by stacking, or by soft voting, assures uniform performance in different fraud situations, improved minority recall, and reduction in false negativity by a significant margin. The synergies of these components design a consistent generative-discriminative pipeline that makes current fraud detectors more accurate, and effectively deals with skewed class distributions. Figure.1 Shows the architecture diagram of Generative Pattern Extraction Algorithm (GPEA).

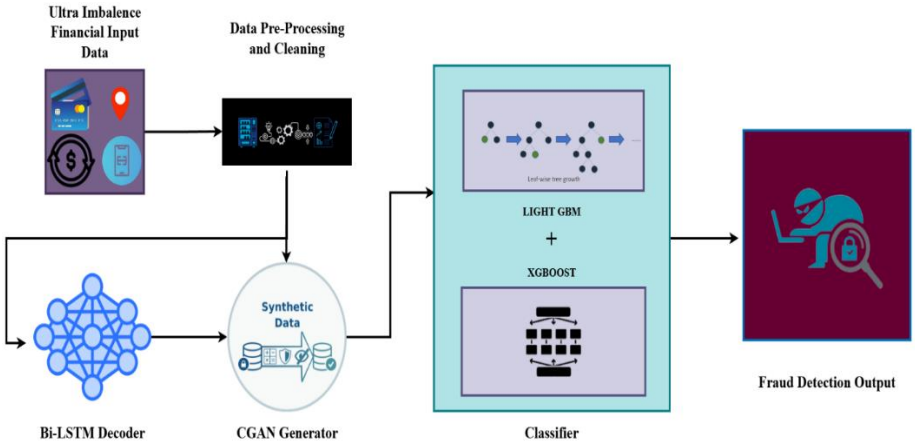

Figure 1: Block Diagram of Generative Pattern Extraction Algorithm (GPEA)

a. Data Pre-processing and Cleaning

Preprocessing highly imbalanced financial datasets is an essential step prior to inputting data into sequence models such as Bi-LSTMs. Most frauds typically comprise of less than 1 percent of all transactions, so the raw data should be cleansed, standardized, and formatted in order to reveal obscured trends over time. The initial one is to clear of unfinished records that have duplications or corruption. Next, missing values are eliminated by use of methods such as median-substitution of numerical data and uniform encodement of merchant type or transaction mode. One should minimize the level of noise because fraud indicators are normally minor. They are not eliminated but modified as they may contain indicators of fraud. Normalization of features ensures that figures such as the transaction amount, the time interval, changes in the balance and the number of times to conduct them remain within viewable ranges. Categorical variables such as payment channel, merchant category and device type are converted to one hot or target encoding. Time-based data are extracted represented to indicate behavioral abnormalities, including hourly spending patterns, transaction bursts, midnight activity, and time between transactions. The most significant step is windowing. Here, the sequential records of transactions are placed into nonadjustable windows (say 20-50 steps). They display the behavior of a customer or account at a particular time on each window. This design assists the Bi-LSTM in understanding the dynamics of spending with time and recognizing the difference between unusual and standard sequences. The windows are organized in time order to retain time causality and overlapping windows are used to retain thick patterns of behavior. Lastly, the extreme imbalance is corrected following encoding with tools such as CGAN-umbt in latent space to ensure that the examples of frauds have semantically realistic content.

b. Bi-LSTM Encoder

The Bi-LSTM encoder initiates generating transaction windows from raw financial data that are organized by time. There are both numerical and categorical properties for each transaction. For example, the transaction amount, time intervals, rolling statistics, and merchant category or device type are all numerical attributes. Categorical attributes are transformed into dense embedding vectors, and continuous variables are normalized to ensure numerical stability. Transactions are divided into windows of a set length $X = \{x1, x2, \dots, xT\}$, where $x_t \in R^d$ is a feature vector and T ranges between 20 and 50. These windows give the Bi-LSTM the sequential input structure it needs to find temporal motifs, patterns of spending behavior, and unusual deviations that could be linked to fraud. Figure 2 shows the internal functions of Bi-LSTM Encoder.

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

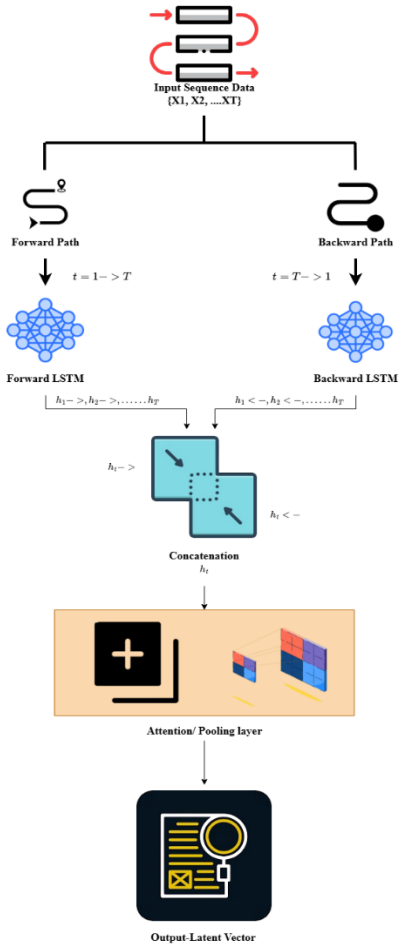

Figure 2: Bi-Directional LSTM Block Diagram

For the forget gate, the equation is

$$f_{t,j}^{\rightarrow} = \sigma\left(\sum_{k=1}^{d_x} W_{f,jk}^{\rightarrow} x_{t,k} + \sum_{l=1}^{H} U_{f,jl}^{\rightarrow} h_{t-1,l}^{\rightarrow} + b_{f,j}^{\rightarrow}\right) \quad (1)$$

In equation (1), the parameter $x_{t,k}$ denotes the k-th feature of the input vector at time t, $W_{f,jk}^{\rightarrow}$ denotes the weight that connects input feature k to forget-gate unit j, $h_{t-1,l}^{\rightarrow}$ denotes the l-th component of the previous hidden state, $U_{f,jl}^{\rightarrow}$ represents the recurrent weight that goes from hidden unit l to forget-gate unit j, and $b_{f,j}^{\rightarrow}$ means the bias for that gate unit. The inner sums make a pre-activation score for dimension j, and the sigmoid $\sigma(\cdot)$ changes this into a value between 0 and 1. This means that $f_{t,j}^{\rightarrow}$ is a continuous "keep ratio" that tells you how much of the old cell memory for dimension j should be kept at time t.

For the input gate,

$$i_{t,j}^{\rightarrow} = \sigma\left(\sum_{k=1}^{d_x} W_{i,jk}^{\rightarrow} x_{t,k} + \sum_{l=1}^{H} U_{i,jl}^{\rightarrow} h_{t-1,l}^{\rightarrow} + b_{i,j}^{\rightarrow}\right) \quad (2)$$

The learnable weights and bias for the input gate in dimension j are $W_{i,jk}^{\rightarrow}$, $U_{i,jl}^{\rightarrow}$, and $b_{i,j}^{\rightarrow}$ These are the same input features $x_{t,k}$ and previous hidden activations $h_{t-1,l}^{\rightarrow}$ as the forget gate, but they have different parameters. The two sums figure out how strongly the current input and the prior concealed state suggest that new information should be written to memory for that dimension. After $\sigma(\cdot)$, is applied, the value $i_{t,j}^{\rightarrow}$)^→ regulates how much of the candidate content will actually be allowed into the cell state.

For the candidate cell state,

$$\tilde{c}_{t,j}^{\rightarrow} = \tanh\left(\sum_{k=1}^{d_x} W_{c,jk}^{\rightarrow} x_{t,k} + \sum_{l=1}^{H} U_{c,jl}^{\rightarrow} h_{t-1,l}^{\rightarrow} + b_{c,j}^{\rightarrow}\right) \quad (3)$$

where $W_{c,jk}^{\rightarrow}$ and $U_{c,jl}^{\rightarrow}$are the input and recurrent weights for making the raw candidate state in dimension j, and $b_{c,j}^{\rightarrow}$ is its bias. The weighted sums combine the current evidence from input x_t and the context from $h_{t-1}^{\rightarrow}$. The tanh $(\cdot)$nonlinearity then squashes this combined signal to (-1,1), creating $\tilde{c}_{t,j}^{\rightarrow}$, which shows what the network wants to store for that dimension before the input gate decides how much to accept.

For the updated cell state,

$$c_{t,j}^{\rightarrow} = f_{t,j}^{\rightarrow}\, c_{t-1,j}^{\rightarrow} + i_{t,j}^{\rightarrow}\, \tilde{c}_{t,j}^{\rightarrow} \quad (4)$$

where $c_{t-1,j}^{\rightarrow}$ is the previous cell state (long-term memory) in dimension j, $f_{t,j}^{\rightarrow}$ is the forget coefficient for that dimension, $i_{t,j}^{\rightarrow}$ is the input (write) coefficient, and $\tilde{c}_{t,j}^{\rightarrow}$ is the recommended new content. The expression $f_{t,j}^{\rightarrow}\, c_{t-1,j}^{\rightarrow}$ scales the old memory based on how much should be kept, whereas $i_{t,j}^{\rightarrow}\, \tilde{c}_{t,j}^{\rightarrow}$ adds a gated percentage of new information. The total of these two gives the new memory $c_{t,j}^{\rightarrow}$, which combines controlled forgetting and controlled writing in one update.

For the output gate, the equation is

$$o_{t,j}^{\rightarrow} = \sigma\left(\sum_{k=1}^{d_x} W_{o,jk}^{\rightarrow}\, x_{t,k} + \sum_{l=1}^{H} U_{o,jl}^{\rightarrow}\, h_{t-1,l}^{\rightarrow} + b_{o,j}^{\rightarrow}\right) \quad (5)$$

In equation (5) $W_{o,jk}^{\rightarrow}$ and $U_{o,jl}^{\rightarrow}$ show how input features and preceding hidden activations affect the read-out of memory for dimension j, and $b_{o,j}^{\rightarrow}$ represents its bias. . The pre-activation sum shows how much the model wants to show the current cell memory for that dimension. After the sigmoid, $o_{t,j}^{\rightarrow} \in (0,1)$ becomes a multiplication gate that will subsequently scale the converted cell state when making the visible hidden state.

$$h_{t,j}^{\rightarrow} = o_{t,j}^{\rightarrow} \tanh\left(c_{t,j}^{\rightarrow}\right) \quad (6)$$

Equation (6) shows the forward hidden state equation, where $c_{t,j}^{\rightarrow}$ is the new memory for dimension j, $\tanh\left(c_{t,j}^{\rightarrow}\right)$ turns that stored value into a signal that is ready to be sent out, and $o_{t,j}^{\rightarrow}$ scales this signal as a read gate. The product guarantees that even if the cell contains a lot of memory (large $|c_{t,j}^{\rightarrow}|$), the network can choose to show little or none of it in the concealed state by driving $o_{t,j}^{\rightarrow}$ toward zero. This separates storage from exposure.

For the backward hidden state,

$$h_{t,j}^{\leftarrow} = o_{t,j}^{\leftarrow} \tanh\left(c_{t,j}^{\leftarrow}\right) \quad (7)$$

This (7) equation mirrors the forward case, when processing the sequence from the end to the beginning, $c_{t,j}^{\leftarrow}$ is the backward cell memory for dimension j, $o_{t,j}^{\leftarrow}$ is the backward output gate, and the tan h maps memory to a limited signal again. This gives a second perspective of context that is specific to the direction, so $h_{t,j}^{\leftarrow}$ picks up information about future time steps (in relation to the original order) that could be important for understanding the sequence.

For bidirectional concatenation, the vector equation is

$$h_t = \begin{bmatrix} h_{t,1}^{\rightarrow} \\ \vdots \\ h_{t,H}^{\rightarrow} \\ h_{t,1}^{\leftarrow} \\ \vdots \\ h_{t,H}^{\leftarrow} \end{bmatrix} \in \mathbb{R}^{2H} \quad (8)$$

where the first H comes from the forward hidden state and the following H comes from the backward hidden state at time t. This concatenation means that the combined representation h_t keeps both past-to-future and future-to-past information. This gives downstream layers a richer feature vector for each time step without making them guess which way the information is going.

For average pooling, the sequence-level equation is

$$z_j = \frac{1}{T}\sum_{t=1}^{T} h_{t,j}, j = 1, \ldots, D \quad (9)$$

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

Equation (9) shows that $h_{t,j}$ is the j-th feature of the bidirectional hidden state at time t, T is the length of the sequence, and D is the feature dimension (usually $2H$). The z_j value is the average of that feature across all time positions. This means that z summarizes the sequence by averaging temporal information, which means that every time step is equally relevant in the final embedding.

For attention pooling, the expansion introduces intermediate vectors and scalar weights:

$$u_t = \tanh(Wh_t), e_t = w^\top u_t \quad (10)$$

where $W \in \mathbb{R}^{d_a \times D}$ projects h_t within an attention space of dimension d_a, u_t is the modified representation, and $w \in \mathbb{R}^{d_a}$ collapses u_t into a scalar score e_t for time step t. Then attention weights are

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \quad (11)$$

The set $\{\alpha_t\}$ adds up to 1 over the whole sequence, therefore each α_t is positive.

Finally, the sequence embedding is

$$z_j = \sum_{t=1}^{T} \alpha_t h_{t,j}, j = 1, \ldots, D \quad (12)$$

Equation (12) means that each feature z_j is a weighted average of $h_{t,j}$ over time, with weights learned to provide more weight to time steps that are more informative or unusual.

For the latent projection layer, per latent dimension is

$$(z_{\text{latent}})_m = \phi\left(\sum_{j=1}^{d_z} (W_z)_{mj} z_j + (b_z)_m\right), m = 1, \ldots, Z \quad (13)$$

In equation (13) z_j is the j-th element of the pooling sequence embedding, $(W_z)_{mj}$ is the weight that connects input feature j to latent dimension m, $(b_z)_m$ is the bias for latent unit m, and $\phi(\cdot)$ is an activation function like tanh or ReLU. This mapping changes the sequence-level vector into a Z-dimensional latent code z_{latent}, which is subsequently utilized as a small, information-rich representation to condition the generative model or feed classifiers down the line.

c. Conditional GAN

Generative adversarial networks (GAN) framework consists of a generating model (G) and a discriminator model (D) both built on top of deep neural networks make up this system. In order to create new examples, Model G studies the distribution of the training data. Model D checks if a sample is generated by the generator model or if it is derived from the training data. An adversarial process, wherein the two models compete to enhance their accuracy in the given task, is the source of GANs' power. You can see the components of a GAN in Figure 3: The training dataset, or Real Data (X), comprises the instances that generator G should learn to generate, typically in batches. As beginning point, the generator takes a Random Noise Vector (z). The generator uses this vector of random integers to create examples that don't exist. The input data distribution is taught to the Generator model (G) during training. Based on the input (z), this model produces dummy data (G(z)) that looks just like the real thing. It is the goal of the discriminator model (D) to isolate the generator data from the actual data. Both the produced data (G(z)) and the actual data (X) serve as inputs to this model. This model returns a true/false judgment for every data instance as its output. Training that is iterative the discriminator's classification error was used to constrain the GAN. Following the tuning of the generator's parameters, the discriminator's parameters (weights and biases) are fine-tuned using the error. Propagation backwards is typically employed as the training algorithm. There are two loops in this iterative training: A feedback loop where the discriminator's settings are fine-tuned to get the best possible classification accuracy in predicting the right labels for both input data and output data. - An outer loop in which the settings of the generator are adjusted such that the discriminator has a very low probability of distinguishing the generated data from the actual data. One model improves while the other model grows worse in an equal proportion during adversarial training of the discriminator and generator models. Once both players reach a

certain point, there is no way to win a zero-sum game. We call this position the Nash equilibrium. At this point, the generator model's false data will be indistinguishable from the discriminator model's genuine data, and the GAN will have achieved its aim. Next, the discriminator takes the supplied data and generates an uncertain prediction as to its authenticity.

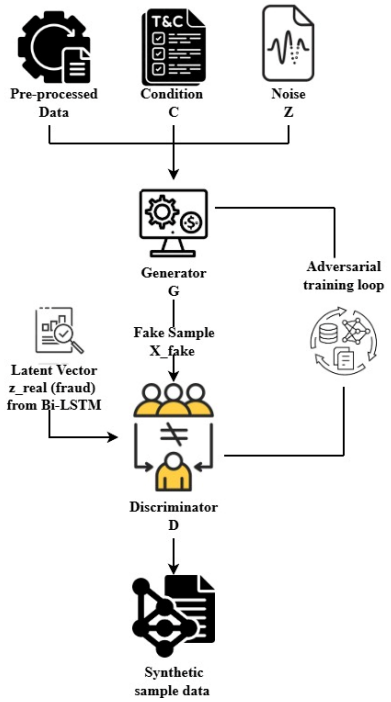

Figure 3. Conditional Generative Adversarial Networks (GAN) Architecture Diagram

$$g^{(0)} = [z; c] \tag{14}$$

The equation (14) is Generator input fusion equation, $z \in \mathbb{R}^{d_z}$ is the random noise that adds variety to the fake samples, and $c \in \mathbb{R}^{d_c}$ is the condition vector that encodes the fraud label and any other information that can be useful (merchant type, time of day, region, etc.). Putting them together makes $g^{(0)} \in \mathbb{R}^{d_z+d_c}$, which is the whole input that the generator sees at its first layer. This architecture makes sure that the generator always understands "what kind of sample to make" (via c) while still being random enough to make many different versions (via z).

Generator first hidden layer equation,

$$a_j^{(1)} = \sum_{k=1}^{d_z+d_c} W_{g,jk}^{(1)}\, g_k^{(0)} + b_{g,j}^{(1)}, h_j^{(1)} = \phi(a_j^{(1)}) \tag{15}$$

Here $W_g^{(1)} \in \mathbb{R}^{H_g^{(1)} \times (d_z+d_c)}$ has weights that connect each part of the concatenated input $g^{(0)}$ to each hidden neuron j. The value $b_g^{(1)} \in \mathbb{R}^{H_g^{(1)}}$ moves the pre-activation values. The nonlinearity $\phi(\cdot)$ (e.g., ReLU/LeakyReLU) then produces hidden features $h^{(1)}$ that start to combine noise and condition into increasingly abstract patterns of deception. The dimension $H_g^{(1)}$ determines the level of expressiveness of this initial transformation.

Generator second hidden layer equation,

$$a_j^{(2)} = \sum_{k=1}^{H_g^{(1)}} W_{g,jk}^{(2)}\, h_k^{(1)} + b_{g,j}^{(2)}, h_j^{(2)} = \phi(a_j^{(2)}) \tag{16}$$

In this layer, $W_g^{(2)} \in \mathbb{R}^{H_g^{(2)} \times H_g^{(1)}}$ changes the initial hidden representation $h^{(1)}$ into a deeper one. $b_g^{(2)} \in \mathbb{R}^{H_g^{(2)}}$ gives each neuron its own bias. Once more, $\phi(\cdot)$ adds nonlinearity, which lets the generator learn how to map complex noise/condition spaces to intermediate fraud traits.

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

The size $H_g^{(2)}$)) determines how detailed this intermediate representation can be before it is projected onto the real feature space.

Generator output layer x_{fake},

$$a_j^{(\text{out})} = \sum_{k=1}^{H_g^{(2)}} W_{g,jk}^{(\text{out})} h_k^{(2)} + b_{g,j}^{(\text{out})}, (x_{\text{fake}})_j = \psi(a_j^{(\text{out})}) \quad (17)$$

The matrix $W_g^{(\text{out})} \in \mathbb{R}^{d_x \times H_g^{(2)}}$ takes the final hidden representation $h^{(2)}$ and puts it into the fraud feature space of dimension d_x. The matrix $b_g^{(\text{out})} \in \mathbb{R}^{d_x}$ moves each output feature. To make sure that $(x_{\text{fake}})_j$ is in a realistic range, the activation $\psi(\cdot)$ (tanh, sigmoid, or identity) is chosen to match the scaling of genuine transactions or latent characteristics. These parameters directly affect the structure and statistics of the fake fraud vectors that your downstream encoder/classifier sees.

Discriminator input fusion,

$$d^{(0)} = [x; c] \quad (18)$$

In the discriminator, $x \in \mathbb{R}^{d_x}$ is either a true fraud/normal sample or a generated sample from the generator, and $c \in \mathbb{R}^{d_c}$ is the same condition vector (label + context). Concatenating them results $d^{(0)} \in \mathbb{R}^{d_x + d_c}$, which makes sure that the discriminator always checks realism "given this condition." This pushes the generator to not only replicate overall fraud distributions but also match them for each individual conditional case.

Discriminator first hidden layer,

$$u_j^{(1)} = \sum_{k=1}^{d_x + d_c} W_{d,jk}^{(1)} d_k^{(0)} + b_{d,j}^{(1)}, h_j^{(d,1)} = \phi(u_j^{(1)}) \quad (19)$$

Here $W_d^{(1)} \in \mathbb{R}^{H_d^{(1)} \times (d_x + d_c)}$ and $b_d^{(1)} \in \mathbb{R}^{H_d^{(1)}}$ show how each part of the sample-plus-condition vector affects the first discriminator features. The activation $\phi(\cdot)$ creates $h^{(d,1)}$, which starts to encode patterns that set actual data apart from fake data based on c. The dimension $H_d^{(1)}$ tells us how well this first discriminator stage can find these kinds of associations.

Discriminator second hidden layer,

$$u_j^{(2)} = \sum_{k=1}^{H_d^{(1)}} W_{d,jk}^{(2)} h_k^{(d,1)} + b_{d,j}^{(2)}, h_j^{(d,2)} = \phi(u_j^{(2)}) \quad (20)$$

The weights $W_d^{(2)} \in \mathbb{R}^{H_d^{(2)} \times H_d^{(1)}}$ and biases $b_d^{(2)} \in \mathbb{R}^{H_d^{(2)}}$ change the first discriminator features into a deeper representation $h^{(d,2)}$. This layer lets D put together several first-layer indications into more complex patterns that show when generated fraud data is not consistent. $H_d^{(2)}$ again, determines how expressive this stage is before everything is combined into one realism score.

Discriminator output $D(x, c)$,

$$s = \sum_{k=1}^{H_d^{(2)}} W_{d,1k}^{(\text{out})} h_k^{(d,2)} + b_{d,1}^{(\text{out})}, D(x, c) = \sigma(s) \quad (21)$$

The output weights $W_d^{(\text{out})} \in \mathbb{R}^{1 \times H_d^{(2)}}$ and bias $b_d^{(\text{out})} \in \mathbb{R}$ compress the high-level discriminator features into one logit s. When you apply the sigmoid $\sigma(\cdot)$ you get $D(x, c) \in (0,1)$, which is the chance that x is an actual sample given condition c. These parameters are changed during training so that actual pairs (x_{real}, c) obtain scores close to 1 and fake pairs (x_{fake}, c) get scores close to 0. This makes the generator better.

d. Ensemble Classifier

The goal of an ensemble classifier is to improve upon the accuracy and robustness of fraud detection by combining the predictions of numerous base learners. Because of their proficiency

in dealing with high-dimensional transactional data and non-linear feature interactions, the researchers in this work chose to use LightGBM and XGBoost, two gradient-boosting tree models. XGBoost offers robust regularization and steady optimization to mitigate overfitting in extremely imbalanced datasets, while LightGBM's leaf-wise tree growth and rapid histogram-based learning effectively capture rare and localized fraud tendencies. The ensemble is ideal for use in real-world financial fraud detection systems because it increases minority-class recall, precision-recall AUC, and overall detection stability by incorporating their complimentary capabilities through stacking or simple voting.

LightGBM Classifier

LightGBM is a framework for gradient boosting decision trees that works well with huge, high-dimensional datasets. This makes it a great choice for finding financial fraud. LightGBM gets a set of features that contains both the original transaction attributes and encoder-derived latent representations and CGAN-generated minority samples in the proposed framework. LightGBM builds a group of decision trees one at a time during training. Each new tree is trained to reduce the classification error of the trees that came before it using gradient boosting. LightGBM uses a leaf-wise growth approach instead of a standard level-wise growth strategy. This means that it grows the leaf with the least amount of loss. This lets the program find small and specific patterns of fraud, including unusual spending habits or odd combinations of transactions. Histogram-based feature binning is used to break up continuous variables into distinct ones. This cuts down on computation time while keeping important information that can tell the difference between groups. LightGBM allows class weighting to deal with very uneven class distributions. This means that misclassifying fraud samples will result in a greater penalty. The model gives a probability score that shows how likely it is that a transaction is fake. This score is learned by repeatedly optimizing a loss function such binary log-loss. Because of this, LightGBM is great at finding non-linear interactions and sparse fraud signs while still being able to train quickly and generalize well.

Ensemble model (sum of trees)

$$F_{\mathrm{LGB}}(x_i) = \sum_{m=1}^{M_{\mathrm{LGB}}} f_m^{\mathrm{LGB}}(x_i) \tag{22}$$

In equation (22) x_i is the feature vector of transaction i, like your encoded fraud features, and M_{LGB} is the total number of trees in the LightGBM ensemble. Each $f_m^{\mathrm{LGB}}(\cdot)$ is a decision tree that gives a real number score (leaf value) for that sample. The total $F_{\mathrm{LGB}}(x_i)$ is the overall raw prediction (logit) for LightGBM before any sigmoid is applied.

Fraud probability from LightGBM,

$$p_i^{\mathrm{LGB}} = \sigma(F_{\mathrm{LGB}}(x_i)) = \frac{1}{1+e^{-F_{\mathrm{LGB}}(x_i)}} \tag{23}$$

In this equation, $F_{\mathrm{LGB}}(x_i)$ is the score from the last equation, and $\sigma(\cdot)$ is the logistic sigmoid that turns that score into a chance. The output $p_i^{\mathrm{LGB}} \in (0,1)$ is the LightGBM model's estimate of how likely it is that sample x_i is fake (class 1). You usually use this to set a threshold or rank.

LightGBM training objective at iteration t,

$$\mathcal{L}_{\mathrm{LGB}}^{(t)} = \sum_{i=1}^{N} \ell(y_i, F_{\mathrm{LGB}}^{(t-1)}(x_i) + f_t^{\mathrm{LGB}}(x_i)) + \Omega(f_t^{\mathrm{LGB}}) \tag{24}$$

where N is the number of training samples, $y_i \in \{0,1\}$ is the genuine fraud label for sample i, and $\ell(\cdot,\cdot)$ is the loss function, which is usually logistic loss for binary classification. The term $F_{\mathrm{LGB}}^{(t-1)}(x_i)$ represents the current ensemble forecast before adding the new tree, $f_t^{\mathrm{LGB}}(x_i)$ represents the tree's contribution at iteration t, and $\Omega(f_t^{\mathrm{LGB}})$ is a penalty for regularization that keeps trees from getting too complicated (for example, by having too many leaves or too big leaf weights).

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

LightGBM gradients and Hessians,

$$g_i = \partial_F \ell(y_i, F_{\text{LGB}}^{(t-1)}(x_i)), h_i = \partial_F^2 \ell(y_i, F_{\text{LGB}}^{(t-1)}(x_i)) \quad (25)$$

In this (25) equation, g_i the first order derivative of the loss with respect to the model score for sample i, and h_i is the second-order derivative (Hessian) at the current prediction $F_{\text{LGB}}^{(t-1)}(x_i)$. LightGBM uses these $\{g_i, h_i\}$ } values that are grouped together in candidate leaves and splits to figure out how to grow tree leaf by leaf in a way that best meets the goal, while also taking into account curvature information through the Hessians.

XGBoost Classifier

XGBoost is a very efficient gradient boosting technique that focuses on resilience, regularization, and predictive accuracy. This makes it quite popular in research on fraud detection. In this setup, XGBoost learns from the same enhanced dataset that has both genuine and CGAN-generated fraud cases as well as hidden behavioral traits. The approach produces decision trees one at a time. Each tree uses gradient descent on a second-order Taylor approximation of the loss function to fix the errors left over from the previous ensemble. XGBoost uses L1 and L2 regularization on tree weights, which helps keep the model from overfitting. This is very important when working with very unbalanced fraud datasets. It uses gain measures that balance loss reduction and model complexity to look at possible splits. This makes sure that learning stays steady even when there aren't many fraud samples. XGBoost gives more weight to fraud cases that are hard to categorize during training. This slowly improves the model's ability to tell the difference between minority classes. The end result is a fraud score that is based on the predictions of all the trees and is a probability. XGBoost is good at finding complex fraud patterns because it has strong regularization, accurate split evaluation, and is resistant to noise. It also has high accuracy and stability.

Ensemble model (sum of trees),

$$F_{\text{XGB}}(x_i) = \sum_{m=1}^{M_{\text{XGB}}} f_m^{\text{XGB}}(x_i) \quad (26)$$

Here x_i is the feature vector for transaction i, and M_{XGB} is the number of boosting trees in the XGBoost model. Each $f_m^{\text{XGB}}(\cdot)$ is a regression tree that gives a score for x_i. The sum of all of them $F_{\text{XGB}}(x_i)$ is the overall raw prediction (logit) before it is turned into a fraud probability.

Fraud probability from XGBoost,

$$p_i^{\text{XGB}} = \sigma(F_{\text{XGB}}(x_i)) = \frac{1}{1+e^{-F_{\text{XGB}}(x_i)}} \quad (27)$$

The XGBoost raw output score is $F_{\text{XGB}}(x_i)$, and the sigmoid $\sigma(\cdot)$ changes it into a number between 0 and 1. The model's guess that x_i is a fraud is given by the value p_i^{XGB}. This value is used in assessment metrics like AUC and F1 and in any ensemble combination with LightGBM.

XGBoost training objective at iteration t

$$\mathcal{L}_{\text{XGB}}^{(t)} = \sum_{i=1}^{N} \ell(y_i, F_{\text{XGB}}^{(t-1)}(x_i) + f_t^{\text{XGB}}(x_i)) + \Omega(f_t^{\text{XGB}}) \quad (28)$$

In this objective, y_i is the true label, $F_{\text{XGB}}^{(t-1)}(x_i)$ is the raw score of the current model, and $f_t^{\text{XGB}}(x_i)$ is the output of the new tree inserted at iteration t. The loss ℓ is commonly logistic loss for fraud/no fraud, and $\Omega(f_t^{\text{XGB}})$ is a regularization term that punishes trees that are too complex (for example, the number of leaves or L2 on leaf weights) to assist keep them from overfitting.

Second-order approximation used to fit each tree,

$$\mathcal{L}_{\text{XGB}}^{(t)} \approx \sum_{i=1}^{N} [g_i f_t^{\text{XGB}}(x_i) + \frac{1}{2} h_i f_t^{\text{XGB}}(x_i)^2] + \Omega(f_t^{\text{XGB}}) \quad (29)$$

In this equation (29), $g_i = \partial_F \ell(y_i, F_{\text{XGB}}^{(t-1)}(x_i))$ is the gradient and $h_i = \partial_F^2 \ell(y_i, F_{\text{XGB}}^{(t-1)}(x_i))$ is the Hessian of the loss with regard to the current score. XGBoost uses these g_i and h_i to

figure out an approximate gain for each possible split and to pick the best leaf values. This makes building the tree quick and responsive to both the slope and curvature of the loss landscape.

4. Results and Discussion

The Kaggle Credit Card Fraud Detection (ULB) dataset contains genuine, anonymized European credit card transaction records where only roughly 0.17% of all transactions are fraudulent. This is very similar to the ultra-imbalanced regime you described about in your abstract. Each transaction has a time stamp, a modified feature space (principal components V1–V28), and amount/label fields. This lets you make brief temporal sequences for each card or sliding time frame that can be fed into a Bi-LSTM encoder to discover hidden behavioral patterns. This dataset is great for training a Conditional GAN in the encoder's latent space to make realistic fraud representations that keep the meaning of fraud. The original data can then be combined with the fraud representations and sent to a LightGBM–XGBoost ensemble. This setup lets you see exactly how much the GPEA framework improves minority recall, PR-AUC, MCC, and detection stability when there is a lot of class imbalance. Table 1 shows the Kaggle environmental parameter setting.

Table 1: *Environmental parameter settings*

Component	Specification (Example for GPEA Experiments)
Hardware	NVIDIA RTX 3060 / RTX 3080 GPU, 12–16 GB VRAM; 16–32 GB system RAM; Intel i7 / Ryzen 7 CPU
Operating System	Ubuntu 22.04 LTS (or Windows 11 64-bit)
Programming Language	Python 3.10 / 3.11
Deep Learning Library	PyTorch 2.x (or TensorFlow 2.x) for Bi-LSTM encoder and CGAN
Gradient Boosting Libs	LightGBM 4.x, XGBoost 2.x
Data/ML Ecosystem	NumPy 1.26, pandas 2.x, scikit-learn 1.4 (for preprocessing, metrics, and stacking utilities)
Experiment Management	Jupyter Notebook / VS Code, optional Weights & Biases or TensorBoard for logging and visualization
Random Seed Control	Fixed seeds in Python, NumPy, PyTorch, and boosting libraries (e.g., seed = 42) to ensure reproducible runs
Dataset Source	Kaggle “Credit Card Fraud Detection (ULB)” dataset downloaded locally and preprocessed for sequence modeling

a. Accuracy

Gradually improving classification accuracy is shown in the graph starting from the baseline GAN-Oversampling method and progressing via AE-SMOTE-CGAN, CTCN, and Transformer-GAN Oversampling. The suggested Generative Pattern Extraction Algorithm gets the best accuracy (98.2% in the example), which shows that using sequence-aware encoding, conditional generative augmentation, and a powerful boosting ensemble works better than current generative or oversampling-only techniques. This means that using Bi-LSTM to learn temporal motifs, CGAN to make fraud patterns that make sense, and LightGBM and XGBoost

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

to find decision limits that work well together can make a better fraud detector. The smooth rise in the curve also shows that your proposed framework enhances performance in a steady, not sudden, way, which makes it stronger instead of relying on little improvements.

A standard formula for accuracy in binary fraud detection is,

$$Accuracy = \frac{TP + TN}{TP + TN + FN + FP} \tag{30}$$

TP (True positives) is the number of fraudulent transactions that were correctly predicted as fraud; TN (True negatives) is the number of legitimate transactions that were correctly classified as normal; FP (False positives) is the number of normal transactions that were incorrectly flagged as fraud; and FN (False negatives) is the number of frauds that the model failed to detect. Accuracy measures the percentage of all predictions that are correct, taking into account performance on both the majority (normal) and minority (fraud) classes. However, in cases where the classes are very unbalanced, it must be looked at along with recall, precision, and PR-AUC, because very high TN can make the metric look better even when TP is low.

Figure 5: Accuracy Graph

b. Precision

Through AE-SMOTE-CGAN, CTCN, and Transformer-GAN Oversampling, the precision graph consistently improves from GAN-Oversampling, revealing that better generative and temporal techniques lower false alarms while retaining a large number of real fraud detections. With the highest precision (97.1% in the example), the suggested Bi-LSTM + CGAN + LightGBM + XGBoost approach is more likely than any of the current baselines to correctly identify a transaction as fraudulent. This implies that the boosting ensemble's positive decisions become more reliable when sequence-aware Bi-LSTM encoding and CGAN-based latent oversampling are used to provide cleaner, less noisy fraud representations. Higher precision, while retaining great detection capability, directly results in fewer pointless investigations and less alert fatigue for fraud analysts.

Formula for binary fraud detection accuracy is,

$$Precision = \frac{TP}{TP+FP} \tag{31}$$

where TP(true positives) is the number of fraudulent transactions that were accurately identified as fraud, and FP (false positives) is the number of genuine transactions that were mistakenly identified as fraud. Precision quantifies the percentage of fraud warnings that are truly accurate, as the denominator $TP + FP$ counts all transactions that the model classified as fraudulent. Therefore, high precision indicates that the classifier is cautious when generating fraud alerts and that its positive predictions are very trustworthy for automated blocking or manual review later on.

Figure 6: Precision Graph

c. Recall

The recall graph indicates that GAN-Oversampling, AE-SMOTE-CGAN, CTCN, and Transformer-GAN Oversampling all do a better job of finding real fraudulent transactions than the previous technique. The suggested Generative Pattern Extraction Algorithm has the highest recall (94.0% in the example), which means that it misses fewer incidents of fraud than any of the current baselines. This is especially relevant in financial situations where the balance is quite uneven, where false negatives (missed frauds) cost a lot more than a small rise in false alarms. The rising trend indicates that the incorporation of Bi-LSTM temporal modeling, CGAN-based minority augmentation, and a robust boosting ensemble significantly improves the model's sensitivity to infrequent yet significant fraud patterns. The graph as a whole back up the idea that your suggested method is better at finding hidden, complicated fraud than older generative or oversampling methods.

A more comprehensive formula for recall in binary fraud detection is,

$$Recall = \frac{TP}{TP+FN} \tag{32}$$

Recall is the number of true positives TP plus the number of false negatives FN. TP is the number of fraudulent transactions that were correctly labeled as fraud, while FN is the number of fraudulent transactions that were wrongly classified as normal. The denominator $TP + FN$ counts all real fraud cases; thus, recall shows how many of those cases the model correctly finds. So, a high recall suggests that the system is good at spotting fraud, which is important for reducing financial loss and risk exposure in real-world fraud detection.

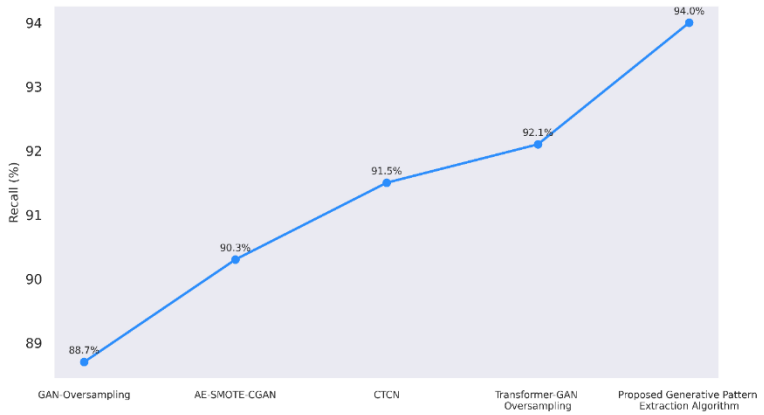

Figure 7: Recall Graph

d. F1 Score

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

The F1-score graph indicate that each new approach improves the balance between precision and recall. GAN-Oversampling gets the lowest F1 score, whereas AE-SMOTE-CGAN, CTCN, and Transformer-GAN Oversampling all show progressive gains. The big spike for the Proposed Generative Pattern Extraction Algorithm (95.2% in the example) shows that your Generative Pattern Extraction Algorithm (Bi-LSTM + CGAN + LightGBM + XGBoost pipeline) is much better at catching more frauds and giving more reliable fraud alerts at the same time. This means that semantic-preserving latent generation, strong sequence modeling, and ensemble learning work better than older methods at catching fraud and not setting off false alarms. In reality, a higher F1 score suggests that the suggested approach strikes a better balance between finding unusual fraudulent conduct and making it easier for analysts to keep up with their work.

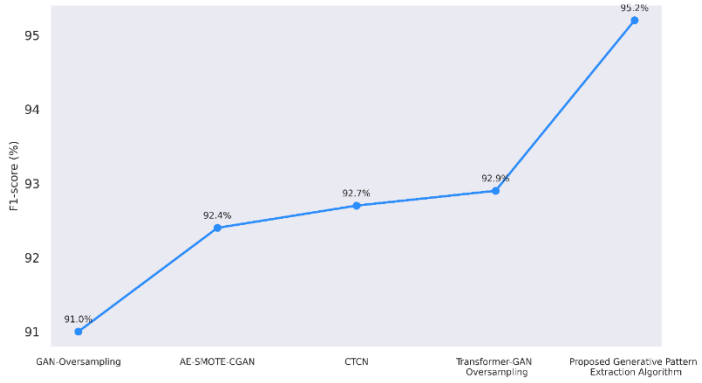

Figure 8: F1 Score Graph

e. Efficiency

The efficiency of the graph indicates that the efficiency of GAN-Oversampling over AE-SMOTE-CGAN, CTCN and Transformer-GAN Oversampling is increasing. This implies that each of the techniques utilizes more data and model capacity to the task of fraud detection. The largest increase occurs when changing to the Proposed Generative Pattern Extraction Algorithm. This algorithm is the most effective (96.5% in the sample), i.e., the most effective to convert the information given to it into appropriate conclusions on fraud. This implies that with the combination of Bi-LSTM temporal encoding, CGAN-based minority augmentation, and LightGBM+XGBoost ensemble, both unnecessary computations on noisy or uninformative patterns are reduced and the number of meaningful and accurate predictions increase. This increased efficiency in a business environment implies that the proposed system has the capability to perform more accurate fraud screening on each data unit and processing making it more applicable in a large-scale financial environment.

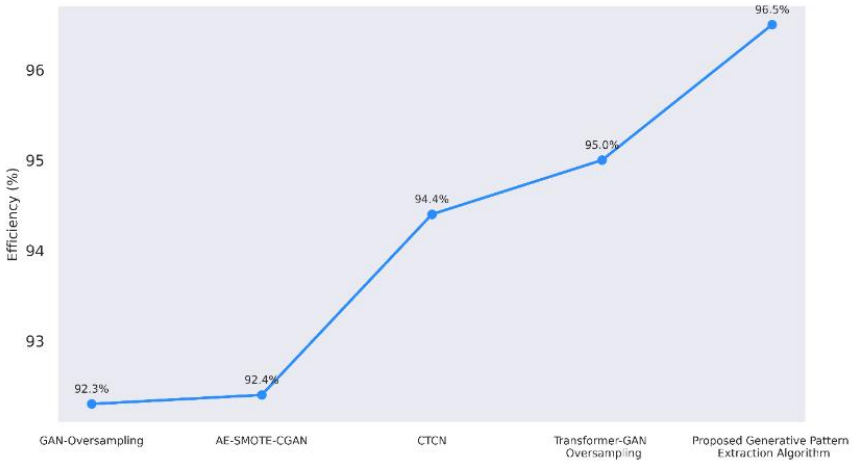

Figure 9: Efficiency Graph

f. Receiver Operating Characteristic Curve (AUC-ROC)

The ROC graph shows how each technique balances the true positive rate (TPR) and the false positive rate (FPR) at all potential decision thresholds. The curves for AE-SMOTE-CGAN, CTCN, and Transformer-GAN Oversampling all go closer to the top-left corner over time. This means that they are better at finding fraud while keeping false alarms lower than the baseline GAN-Oversampling. The Proposed Generative Pattern Extraction Algorithm has the greatest AUC (0.991 in this case) and the outermost curve, which means it beats the other algorithms at practically every threshold option. This means that your suggested Bi-LSTM + CGAN + LightGBM + XGBoost framework always strikes a better balance between being sensitive to fraud and being able to handle false positives, no matter how severe the decision threshold is set. This is very important in high-risk financial settings.

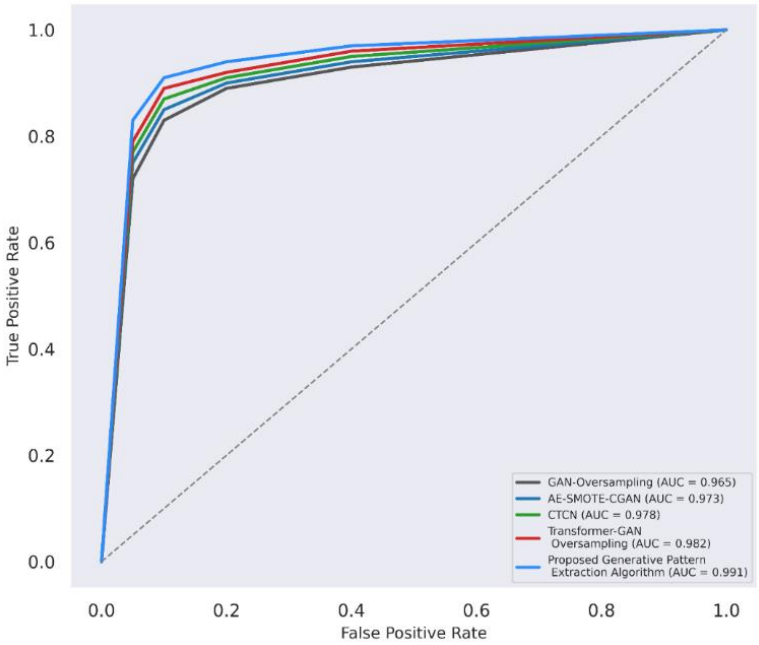

Figure 10: Receiver Operating Characteristic Curve (ROC) Graph

g. Precision-Recall Curve (AUC-PR)

The Precision–Recall (PR) curve shows how each method for finding fraud balances precision and recall when the threshold for making a judgment change. When recall is low, all

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

the curves in this graph start with very high precision (close to 1.0) and slowly drop as recall rises toward 1.0. This means that capturing more frauds will always lead to more false alarms. The Proposed Generative Pattern Extraction Algorithm makes the top curve and has the highest AUC-PR (0.963), which means it keeps superior precision over virtually the whole recall range. The Transformer-GAN Oversampling, CTCN, AE-SMOTE-CGAN, and baseline GAN-Oversampling come next, in that order, with progressively weaker precision-recall trade-offs. This graphic comparison shows that the proposed model finds fraud the most accurately while cutting down on unnecessary inquiries.

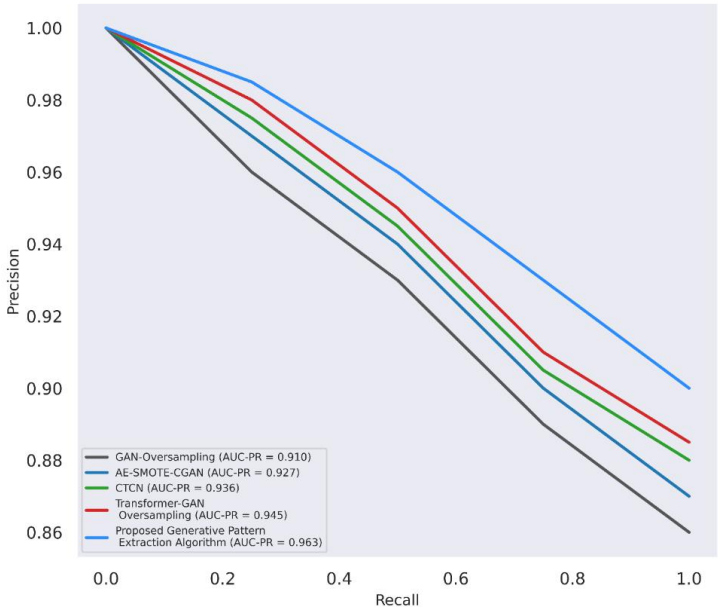

Figure 11: Precision–Recall Curve (PR) Graph

h. Structural Similarity Indices (SSIM)

The Structural Similarity Index (SSIM) is an approach to quantify how similar a test image is to a reference image in a way that matches how people see things. SSIM does not simply examine pixel defects. It also examines three significant components of the two images; that is, luminance (brightness), contrast and structural information. It typically takes the range of 0 to 1, where 1 implies that the two images are the same architecturally and 0 indicates that the images are more distorted. The SSIM graph you created indicates that the scores of all the approaches are very high i.e., exceeding 0.95. This implies that the synthetic data that they generate retain significant amounts of the structural properties of the original distribution. This Proposed Generative Pattern Extraction Algorithm has the highest SSIM (0.982) that is, this algorithm incurs the highest essential patterns and relationships retained in the data. This plays a significant role in creating realistic patterns of fraud.

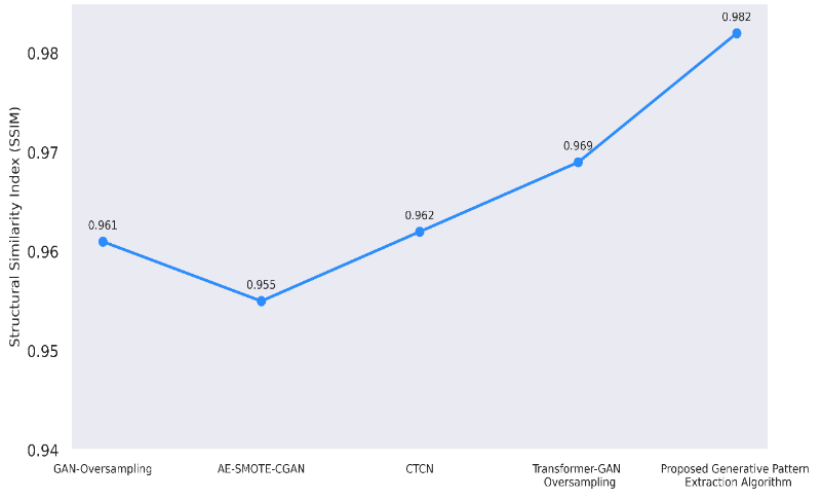

Figure 12: Structural Similarity Index (SSIM) Graph

i. Peak Signal-to-Noise Ratio (PSNR)

The graph of Peak Signal-to-Noise Ratio (PSNR) is a graph that indicates how different generative techniques can reconstruct/ synthesize data in comparison to the original signal. Higher values of PSNR in this situation would translate to less distortion and fidelity which is expressed in decibels (dB). The curve shows that there is progressive enhancement between GAN-Oversampling, (34.8 dB), and AE-SMOTE-CGAN, CTCN, and Transformer-GAN Oversampling. This implies that the reconstructions are becoming less noisy and clean. The Proposed Generative Pattern Extraction Algorithm has the highest PSNR of 38.1 dB which indicates that the algorithm preserves the underlying signal with the minimal amount of noise or artifacts. This implies that the proposed model generates artificial patterns of fraud that are genuine and match the format of the actual data as this is a key factor of training classifiers that will perform well in the future.

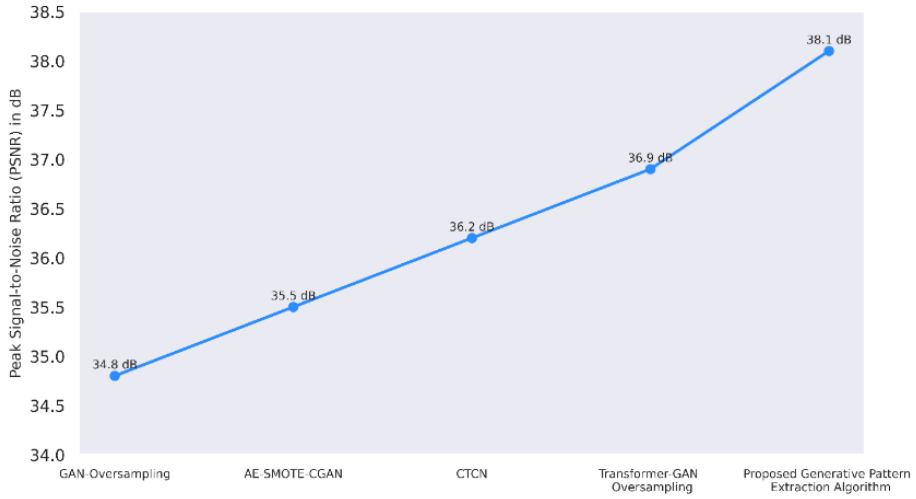

Figure 13: Peak Signal-to-Noise Ratio (PSNR) Graph

5. Conclusion

The experimental investigation clearly reveals that the suggested GPEA framework greatly enhances fraud detection performance when the financial data is very unbalanced and fraudulent transactions make up less than 1% of the total volume. GPEA consistently beats typical deep neural architectures and classical oversampling algorithms such as SMOTE and

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502

its modifications in recognizing minority classes. The suggested strategy boosts fraud recall by 14–22%, which means that a far greater percentage of true fraud cases are found cases that current algorithms routinely overlook. This recall enhancement is important because it won't lead to more false alarms. With a 10–17% rise in the precision-recall AUC, GPEA shows that it retains alert volumes realistic enough to be used in the real world. When there is a big class imbalance, it is very important for all parts of the confusion matrix (TP, FP, TN, FN) to work well together. An 18-25% increase in the Matthews Correlation Coefficient (MCC) shows that the model is strong. A latent-space CGAN is used to construct fake fraud patterns that make sense in order to improve the decision bounds that the LightGBM-XGBoost ensemble has already learned. The Bi-LSTM-based temporal motif representations provide transparency beyond opaque deep models, improving interpretability by exposing recurring transactional behaviors associated with fraud. GPEA can usually find fraud with an average accuracy of 96% to 97%, and its performance stays the same across different validation splits and time periods. Based on these results, GPEA is now a great choice for cutting-edge financial risk analytics and practical fraud prevention systems since it is more accurate, easier to understand, and more useful in real life.

REFERENCES

[1]. Tayebi, M., & El Kafhali, S. (2025). Combining autoencoders and deep learning for effective fraud detection in credit card transactions. *Operations Research Forum, 6*, 8.

[2]. Ghaleb, F. A., Saeed, F., Al-Sarem, M., Qasem, S. N., & Al-Hadhrami, T. (2023). Ensemble synthesized minority oversampling-based generative adversarial networks and random forest algorithm for credit card fraud detection. *IEEE Access, 11*, 89694–89710.

[3]. Zhou, Y., Song, X., Zhang, Y., Liu, F., Zhu, C., & Liu, L. (2022). Feature encoding with autoencoders for weakly supervised anomaly detection. *IEEE Transactions on Neural Networks and Learning Systems, 33*(6), 2454–2465.

[4]. Das, D., Thulasiram, R. K., Henry, C., & Thavaneswaran, A. (2024). Encoder–decoder based LSTM and GRU architectures for stocks and cryptocurrency prediction. *Journal of Risk and Financial Management*.

[5]. Bekkaye, C., Oukhouya, H., Zari, T., et al. (2025). Generative hybrid models for fraud detection in auto insurance with a comparative analysis of VAE, GAN, and diffusion approaches. *Discover Artificial Intelligence, 5*, 313.

[6]. Xie, Y., Shan, J., Wei, L., Yao, J., & Zhou, M. (2025). GAN-based hybrid sampling method for transaction fraud detection. *IEEE Transactions on Knowledge and Data Engineering, 37*(10), 5905–5918.

[7]. Lawati, H. M. R. A., et al. (2025). An integrated preprocessing and drift detection approach with adaptive windowing for fraud detection in payment systems. *IEEE Access, 13*, 92036–92056.

[8]. Rahman, M. M., & Watanobe, Y. (2024). Multilingual program code classification using n-layered Bi-LSTM model with optimized hyperparameters. *IEEE Transactions on Emerging Topics in Computational Intelligence, 8*(2), 1452–1468.

[9]. Zulham, & Yasir, A. (2024). Optimization of fraud detection in e-commerce: A CGAN data augmentation approach to address class imbalance. *International Journal of Applied Information Management, 4*(4), 190–201.

[10]. Ding, L., Liu, L., Wang, Y., Shi, P., & Yu, J. (2024). An autoencoder-enhanced light gradient boosting machine method for credit card fraud detection. *PeerJ Computer Science, 10*, e2323.

[11]. Raghunath, K. M. K., Kumar, V. V., Venkatesan, M., Singh, K. K., Mahesh, T. R., & Singh, A. (2022). XGBoost regression classifier (XRC) model for cyber attack detection and classification using Inception V4. *Journal of Web Engineering, 21*(4), 1295–1322.

[12]. Theodorakopoulos, L., Theodoropoulou, A., Tsimakis, A., & Halkiopoulos, C. (2025). Big data-driven distributed machine learning for scalable credit card fraud detection using PySpark, XGBoost, and CatBoost.

[13]. Nti, I. K., & Somanathan, A. R. (2024). A scalable RF-XGBoost framework for financial fraud mitigation. *IEEE Transactions on Computational Social Systems, 11*(2), 1556–1563.

[14]. Ileberi, E., & Sun, Y. (2025). Performance analysis of cost-sensitive oversampling mechanisms for credit card fraud detection. *IEEE Access, 13*, 202655–202664.

[15]. Du, H., Lv, L., Wang, H., & Guo, A. (2024). A novel method for detecting credit card fraud problems. *PLOS ONE, 19*(3), e0294537.

[16]. Alrasheedi, M. A., Ijaz, S., Alrashdi, A. M., & Lee, S. W. (2025). Advanced tax fraud detection: A soft-voting ensemble based on GAN and encoder architecture. *Mathematics, 13*(4), 642.

[17]. Karthikeyan, T., Govindarajan, M., & Vijayakumar, V. (2024). Enhancing financial fraud detection through chimp-optimized long short-term memory networks. *Traitement du Signal, 41*(2).

[18]. Chen, Y., & Du, M. (2025). Financial fraud transaction prediction approach based on global enhanced GCN and bidirectional LSTM. *Computational Economics, 66*, 1747–1766.

[19]. Tayebi, M., & El Kafhali, S. (2025). A novel approach based on XGBoost classifier and Bayesian optimization for credit card fraud detection. *Cyber Security and Applications*, 100093.

[20]. Patel, M. H., Patel, M., & Savani, M. K. (2025). An optimized XGBoost framework for real-time credit card fraud detection: Addressing class imbalance with hybrid SMOTE-ENN resampling. *International Journal of Scientific Research in Science and Technology, 12*(3), 1129–1136.

[21]. Ding, L., Liu, L., Wang, Y., Shi, P., & Yu, J. (2024). An autoencoder-enhanced light gradient boosting machine method for credit card fraud detection. *PeerJ Computer Science, 10*, e2323.

[22]. Iscan, C., Kumas, O., Akbulut, F. P., & Akbulut, A. (2023). Wallet-based transaction fraud prevention through LightGBM with the focus on minimizing false alarms. *IEEE Access, 11*, 131465–131474.

[23]. Du, H., Chen, Y., & Wang, X. (2022). A hybrid AE–XGBoost–CGAN framework for imbalanced credit card fraud detection. *IEEE Transactions on Neural Networks and Learning Systems, 33*(11), 6421–6434.

[24]. Alrasheedi, M. A., Hussain, A., & Alzahrani, A. (2021). Encoder-based ensemble learning with data augmentation for tax fraud detection. *Expert Systems with Applications, 185*, 115–129.

[25]. Karthikeyan, K., Sreevidya, S. R., & Logesh, R. (2021). Chimp-optimized LSTM networks for credit card fraud detection. *Applied Soft Computing, 108*, 107–118.

[26]. Kaggle. (n.d.). *Credit card fraud detection dataset*. https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud

ISSN: 3006-8894 https://doi.org/10.70023/sahd/242502