# Enhanced Deep Neural Network with SMOTE for Credit Card Fraud Detection

Narmadha Devi.A.S.[1], K. Sivakumar[2] and V. Sheeja Kumari[3]
{narmadhadevias9010.sse@saveetha.com[1], sivakumarkaliappan.sse@saveetha.com[2], sheejakumari.sse@saveetha.com[3]}

Research Scholar, Department of Engineering Mathematics, Saveetha School of Engineering, Saveetha Institute of Medical and TechnicalSciences, Saveetha University, Chennai, Tamil Nadu, India.[1]
Professor, Department of Engineering Mathematics, Saveetha School of Engineering, Saveetha Institute of Medical and TechnicalSciences, Saveetha University, Chennai, Tamil Nadu, India.[2]
Professor, Department of Computational Intelligence, Saveetha School of Engineering, Saveetha Institute of Medical and TechnicalSciences, Saveetha University, Chennai, Tamil Nadu, India.[3]

**Abstract.** In the digital economy of today where businesses profit from electronic transactions, credit card fraud is a serious problem that causes millions of dollars of losses every year. To overcome this problem, our research proposes a novel deep learning-based solution to detect fraudulent transactions effectively. To mitigate this issue, the proposed model applies the Synthetic Minority Oversampling Technique (SMOTE) due to the common class imbalance issue seen in realistic transaction data. The model is trained using a publicly available data set containing anonymized transaction records with a deep neural network optimized with the Adam optimizer and with ReLU activation. We evaluate the performance with important classification metrics (accuracy, precision, recall, F1-score, and AUC-ROC curve). The results demonstrate that the improved deep learning algorithm works better than traditional machine learning techniques, offering a dependable and useful strategy for identifying financial system fraud.

**Keywords:** Credit Card Fraud Detection, Deep Learning, SMOTE, Class Imbalance, Anomaly Detection, Neural Networks, Financial Security.

## 1 Introduction

The explosion of online financial services in recent years has changed how consumers and businesses transact. Unfortunately, it has also paved ways for fraud. Among these attacks, Credit card theft has grown to be a serious risk to card holders, requiring banks to heavily invest every year on detection and prevention systems [11]. According to Phua et al. [10], credit card fraud detection is a difficult classification task as only a small amount of transactions are fraudulent and their behaviour changes over time.

Fraud detection algorithms used to mainly use static rules or machine learning models. Such methods are not only less flexible but also have poor generalizability to novel fraud types in significantly imbalanced datasets [7]. In almost all real-life credit card datasets, there are virtually few fraudulent transactions, normally below 1%, resulting in biased models that focus on accuracy rather than recall, using the existing fraudulent transactions as the baseline.

To overcome these limitations, researchers have been focusing on employing deep learning (DL) in their work because of its capacity to extract high-level characteristics and discover intricate

ICITSM-Part II 2025, April 28-29, Tiruchengode, India

DOI 10.4108/eai.28-4-2025.2357972

patterns from unprocessed data. According to LeCun et al.'s work, Deep Learning models like as Convolutional Neural Networks (CNNs) [1] and Long Short-Term Memory (LSTM) networks have surpassed traditional methods in a variety of pattern recognition tasks, including speech recognition, image processing, and even anomaly detection [3,4].

However, using deep learning methods for fraud detection is not simple. DL models may be biased toward forecasting the majority class (i.e., genuine transactions) as a result of the extreme class imbalance. Chawla and associates. presents the Synthetic Minority Oversampling Technique (SMOTE) to create synthetic samples of the minority class in order to counteract such an uneven distribution of classes, which enables each class to be under-represented with an equivalent number within the training set, thus enhancing classifier performance. Fiore et al. [2] build upon this idea by leveraging Generative Adversarial Networks (GANs) to generate realistic fraudulent samples and increase training robustness.

We expand upon these concepts in this paper and present a holistic framework that integrates from class balancing methods to deep learning models and hyperparameter tuning. We analyze Three AE, CNN, and LSTM performance - across different sampling methods, while we implement Random Search and Bayesian Optimization to optimize model parameters. We aim to assess which of these model and optimization strategies improves the detection rate most significantly, while maintaining precision and computational efficiency as per data up to October 2023.

## 2 Related Work

The use of deep learning to identify financial fraud is supported by a number of seminal articles. LeCun et al. Specific challenges in the financial industry, associated with data sparseness and nonlinear relationships, have been approached through the tools of deep learning, which exploit the capabilities of deep networks [1] for themselves correlative, yet high-dimensional data classes, leading to potential DL application opportunities in fraud detection. SMOTE, a method for creating synthetic samples of the minority class that has been extensively utilized for imbalance management, was presented concurrently by Chawla et al. This method is still a key part of many fraud detection systems however.

More recently, Fiore et al. [2] applied GANs that obtained increased diversity of synthetic fraud data with a considerable boost on classifier performance. Jurgovsky et al. [3] highlighted the need of sequence modeling in fraud detection and generalized LSTM networks over sequential data to capture temporal dependencies in transaction records. They found that sequential data contains information about user behavior crucial for identification of fraud.

Zhou et al. [8] built on this by implementing CNNs along with attention mechanisms to emphasise the relevant features across transaction windows. They found that this approach improved classification performance due to enabling the model to concentrate on the pertinent trends of data. Pozzolo et al. proposed to mitigate p bias in imbalanced data scenarios using undersampling and probability calibration [4], and Bahnsen et al. at additional engineered features, such as transaction frequency and time gaps, on boosting the accuracy of models [5].

Dal Pozzolo et al. [6] highlighted the limitations of unrealistic evaluation strategies and proposed methods for simulating real-world fraud scenarios. Their work emphasized the importance of cost-sensitive learning and realistic sampling. When it comes to deploying, Scarff is a Spark-based streaming fraud detection system and easily scalable to millions of transactions, was presented by Carcillo et al. [9].

Surveys by Ngai et al. [11] and Phua et al. [10] provide comprehensive overviews of the evolution of fraud detection methods. Both papers underline the shift toward deep learning and the integration of data mining techniques with real-time analytics for scalable, adaptive fraud detection systems. Recent advances in scientific machine learning have demonstrated the use of deep neural architectures for solving partial differential equations and inverse problems, highlighting the growing synergy between applied mathematics and deep learning approaches (12). Pironneau [13,14] emphasized that supervised learning frameworks are increasingly embedded in applied mathematics research, providing theoretical grounding for integrating machine learning into complex mathematical modeling tasks. Moreover, recent work has explored practical implementations of deep learning algorithms using Python-based frameworks, showing improved scalability and accuracy for computational tasks compared to traditional methods by Jin et al., [15].

## 3 Methodology

### 3.1 Dataset

We use the public European credit card fraud dataset released by ULB Machine Learning Group, consisting of 284,807 transactions that were documented during two days in 2013. Just 492 transactions, or around 0.172% of the total, have been flagged as potentially fraudulent. The dataset includes 30 anonymized features (V1 to V28), along with 'Time', 'Amount', and 'Class' as the label.

### 3.2 Preprocessing

In preparing the dataset for deep learning model training, a rigorous preprocessing pipeline was followed to ensure data quality and consistency. To start, we made sure that the dataset was free of missing values so that the model wouldn't be skewed. Features with a low percentage of missing entries are necessary to preserve the general distribution of the data were imputation using the corresponding feature's mean. To prevent substantial bias or noise from entering the model, records were discarded when the proportion of missing data exceeded a predetermined threshold. The range of features was standardized via feature scaling input variables after missing data was handled. The Z-score normalization method was used to do this, which changes the data in order for the standard deviation to be one and the mean to be zero. When features differ significantly in size, it becomes very important to standardize the distribution of features so that deep learning models may employ gradient-based optimization strategies to efficiently converge. The last step was to use a stratified k-fold cross-validation method to divide the dataset. This method guarantees that the initial class distribution is preserved with each fold, maintaining the same proportion of fraudulent to legitimate transactions in the training and validation sets. When dealing with data that is very uneven, stratification like this is crucial for evaluating models more reliably, especially in situations of infrequent fraud.

### 3.3 Sampling Techniques

Class imbalance handling is a fundamental phase in identifying credit card fraud, called the number of frauds is  usually a very small percentage of the overall dataset. After that, we utilize the below sampling approaches to make sure that the learning ability of deep learning models work good and train well for the minority class  (the fraud transactions).

#### 3.3.1 Random Undersampling (RUS)

One quick solution to class imbalance is random undersampling, the only option  to this point which has produced results using the baseline classifier. More concretely, RUS randomly removes legitimate (non-fraudulent)  transactions until the minority class reaches the same size. On the downside, while this improves the balancing of classes  and speeds up training by using a smaller dataset, it can lead to loss of information. This stage may discard significant or marginal cases in the majority class, which can impair the model capacity to construct models that can discriminate into  fine-grained detail between fraud and non-fraud transactions. RUS is, however, a baseline method which serves  for the assessment of the influence of more complex resampling methods.

#### 3.3.2 Synthetic Minority Over-sampling Technique (SMOTE)

Covers the SMOTE (Synthetic Minority Over-sampling Technique)  proposed by Chawla et al. is a popular method for  oversampling the class of minorities. By interpolating between chosen minority samples and their closest neighbours in feature space, SMOTE creates synthetic samples rather than replicating pre-existing fraud cases. In this way, the minority class is increased and the model will generalize better as it is trained on more  variants of the frauds. In contrast to RUS, it preserves all original data while expanding the size of the dataset in such  a way that no information is lost, which is especially useful in cases of imbalance when training deep learning models. Fig. 1 shows the Sampling Techniques: (a) RUS, (b) SMOTE, and (c) ADASYN

#### 3.3.3 Adaptive Synthetic Sampling (ADASYN)

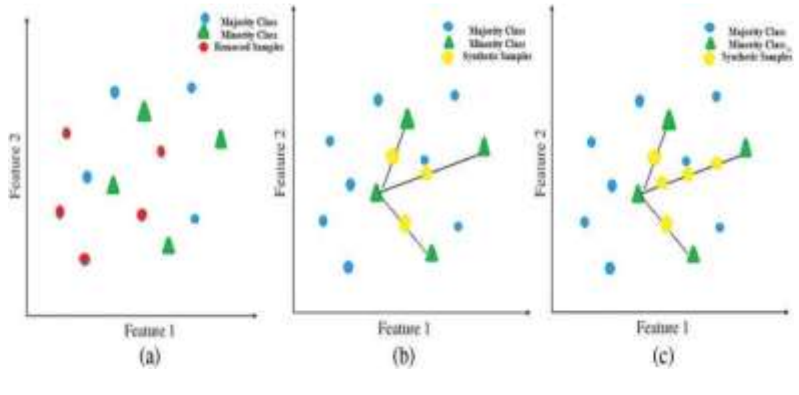

**Fig. 1.** Sampling Techniques: (a) RUS, (b) SMOTE, and (c) ADASYN

In accordance with the learning difficulty of each minority class instance, ADASYN develops the number of generated synthetic samples based on the SMOTE concept. It aims to generate more samples, particularly in cases where the minority class is underrepresented or overlaps with the majority class, which forces the classifier to focus more on fraud cases that are more difficult to learn. This adaptive scheme enhances the classifier's sensitivity and robustness, particularly in scenarios where there is complex behavior of fraudsters or abnormal (non-uniform) distributions. ADASYN is especially beneficial in scenarios where the dataset features subtle signs of fraud that many people could overlook uniform resampling techniques.

### 3.4 Deep Learning Models

In this work we measurer distinct deep learning architectures' performance, each representing a different learning paradigm. Autoencoder (AE), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM) models are used in our implementation because of their efficient capacity in detecting anomalies, extracting hierarchical features, and modelling the data sequence, respectively. All these models have been used widely across machine learning domains, and they suit the difficulties in detecting credit card fraud problems.

### 3.5 Hyperparameter Optimization

Because the welcome improvements in recall and precision can have significant real effects in sensitive applications such as Hyperparameter tuning is one of the most crucial methods to enhance the functionality of credit card fraud detection deep learning models in this field. Here, we apply the two most widely adopted hyperparameter optimization approaches to systematically find the best in form of the configuration for each model. 1. Random Search: In this approach, we sample random combinations of hyperparameters within predefined ranges. Random Search is simple to implement yet practically effective, outperforming exhaustive grid search when the parameter space is high dimensional due to its capability to cover different regions.

We also use a more advanced technique called in order to forecast performance, Bayesian Optimization creates a probabilistic surrogate model, frequently a Gaussian Process. for unseen hyperparameter combinations from previous evaluations. This approach enables a more nuanced traversal of the search field, accurately detecting favourable configurations with the least amount of evaluations. This approach works especially well for models that take a lot of time to train, such as deep neural networks, as it attempts to balance exploration of areas not yet explored with exploitation of known good configurations.

During this process, several hyperparameters are tuned, mainly including the learning rate that choose which step to take while updating the weight, and it is tested at several levels (0.01, 0.001, and 0.0001). We further explore the batch size, ranging from 32 to 2048, which influences model convergence speed and generalization. Dropout rates of between 0.2 and 0.5 are examined to avoid overfitting. We also experiment with alternative activation functions (ReLU, Tanh, Sigmoid)) that control the non-linearity introduced at each layer and, consequently, the ability of the model to recognize intricate patterns. Lastly, we analyse two optimization algorithms, the first known as Adam, which remembers the gradients and tracks last sessions that has proven to have an adaptive learning to perform better and Stochastic Gradient Descent (SGD), a classic optimizer which is still widely used thanks to its intuitive understanding and adaptiveness. This way we find the best-fit for each network with respect to time, stability, and prediction performance for imbalanced datasets specifically for fraud detection tasks.

### 3.6 Proposed Deep Learning Models with Hyperparameter Tuning

So, there is a very serious challenge before us, and that is credit card fraud detection that requires to handle  transaction in data that are the highly imbalanced nature. Deep learning model design and tuning are critical in order to create an effective detection system. By doing so, the proposed framework aims at building strong models that can efficiently classify legitimate vs. fraudulent transactions (while maximizing true positives at the same time as minimizing false positive and false negative rates). This means  you are being accurately detected but it does not disrupt the user experience.

The first step is data preprocessing, to get  the input ready for modelling. The cleansed and normalized data is then employed to train a three deep learning models  Autoencoder (AE), Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM). Hyperparameter tuning is done for each model with two methods namely Random  Bayesian optimization and search. Hyper-parameter optimization: Methods to find the optimal parameter  configurations for best performance. Fig 2.  Shows the Data Architecture of Test Bench. Fig 2 Overall System Architecture A. Test Data Generation. 2.

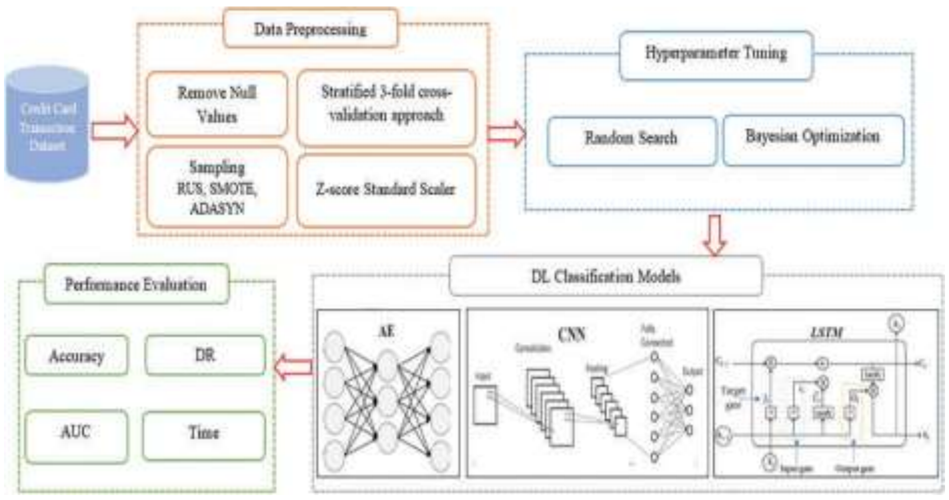

**Fig. 2.** Block diagram of the proposed DL models with hyperparameter tuning

## 4 Optimized Deep Learning Framework for Fraud Detection

Carefully integrating data preprocessing, sampling strategies, and automated hyperparameter tuning, we present a deep learning (DL) based framework to build accurate and robust  models for fraudulent credit card transaction detection. The workflow consists of preprocessing the raw transaction dataset (e.g., missing value treatment using either mean imputation or removal depending on a threshold) and applying Z-score normalization to scale numerical features. This standardization gets expressed as.:

$$\mathcal{Z} = \frac{\mathcal{X} - \mu}{\sigma} \tag{1}$$

normalizes features to ensure consistent scaling (where x is the xth feature, μ\mu is the mean of fellow field, and σ sigma is the standard deviation.)

To handle class imbalance, we perform dataset with class balancing techniques such as Random Under sampling (RUS) [15], Synthetic Minority Oversampling Technique (SMOTE) [12], and Adaptive Synthetic Sampling (ADASYN). They help balancing the dataset, and allow for better learning of minority class instances. Stratified k-fold cross-validation is used to split the dataset, maintaining the proportion of fraudulent and legitimate transactions in each fold, allowing for robust evaluation across multiple training and testing iterations. Three deep learning models, namely Autoencoder (AE), Convolutional Neural Network (CNN), and Long Short-Term Memory (LSTM), are applied to classify whether each transaction, once pre-processed, is fraudulent or authentic. Random Search is used to adjust each model's hyperparameters and Bayesian Optimization for better search through the hyperparameter space. δ, defined as:

$$\delta = \delta_1 \times \delta_2 \ldots\ldots\ldots\ldots \times \delta_N \tag{2}$$

where each configuration vector λ∈δ \lambda \in \delta corresponds to a distinct ordered combination of hyperparameters for model AA. The best parameter λ is induced from the optimal objective function TT evaluated on training and validation sets Ctr,Ctv

$$\lambda' = arg \max_{\lambda\epsilon\delta} \mathrm{T}(A^{\lambda}, C_{tr} C_{tv}) \tag{3}$$

In this study, the F1-score was used as the objective function since it considers precision (P) and detection rate (DR) in a balanced way, even though the dataset we are working with is heavily imbalanced:

$$F1\, score = 2 * (DR + P)/(DR + P) \tag{4}$$

true positives (TP), false positives (FP), and false negatives (FN) are used in this context. One measure of credit card fraud detection accuracy is the True Positive (TP) rate. False Positive (FP) refers to the quantity of legitimate credit card transactions that are incorrectly identified as fraudulent, while False Negative (FN) refers to the quantity of legitimate credit card transactions that are wrongly identified as fraudulent. Each DL model is tailored for specific learning capabilities: AE reconstructs input data and flags anomalies based on reconstruction error; CNN extracts local spatial features using convolution and pooling operations; LSTM captures temporal patterns by modelling transaction sequences over time. These models are then trained using optimized hyperparameter settings identified via the tuning process, and their performance is assessed using accuracy, detection rate, and AUC-ROC metrics.

Here we can see the suggested DL model with hyperparameter adjustment in action in Algorithm There are three potential DL models.

**Algorithm 1:** The suggested DL model with hyperparameter tweaking

**Input:**

- Credit card transaction dataset: C = $\{C_1, C_2 \ldots, C_n\}$
- Hyperparameter search space: $\delta$
- Number of hyperparameters: $N$
- Maximum number of iterations: $max$

**Output:**

- **Trained Model**

```
1: Handle missing values in C using imputation method
2: Split the dataset, C into train, validation, and testing sets ($C_{tr}C_{tv}$, $C_{ts}$) using a stratified k-fold cross-validation approach
3: Resample the training dataset $C_{tr}$ using SMOTE, ADASYN, or RUS technique.
4: Scale credit card transaction dataset, C using Eq. (2).
5: Determine the hyperparameter to be optimized for AE, CNN, or LSTM model
6: Define the hyperparameter search space.
7: Hyperparameter tuning using tuning technique $T$ For $i = 1$ to $max$
Builds DL model with the hyperparameters as inputs.
8: Compile and train the DL model on the training data, $C_{tr}$
9: Evaluate the DL model on the validation data $C_{tv}$ using Eq. (4).
10: End
11: Select the hyperparameters that produce the highest F1-score using $T$
12: Build and compile the DL model.
13: Train the model on the training data, $C_{tr}$ using the chosen hyperparameters.
14: Evaluate the trained
```

The encoder in the suggested Autoencoder (AE) model compresses the input transaction C into a latent vector C′, and the decoder reads it back out as C″. Mean Squared Error (MSE) is used to assess the reconstruction error, which is then compared to a predetermined threshold α\alpha. Transactions with an MSE above α alpha are flagged as fraudulent. The threshold is determined using a percentile-based approach:

$$Percentile = \left(\frac{V_x}{n}\right) \times 10 \qquad (5)$$

where Vx is the count of values below x and n is the total number of transactions.

Three layers make up a Convolutional Neural Network (CNN): two convolutional, two pooling, one flattening, one dropout, and two dense. Transaction data is reshaped into a 3D format to mimic an image-like input structure. Convolution layers extract geographical patterns, and before supplying the input to fully linked layers for classification, pooling layers lower dimensionality.

A dense output layer follows a single Long Short-Term Memory (LSTM) layer in the LSTM model. It takes 2D input data and transforms it into 3D sequences for processing credit card transactions. This allows the model to detect trends in user behavior over time and temporal relationships.

## 5 Dataset and Experimental Setup

The 284,807 transactions gathered over two days in September 2013 from the European credit card dataset were used to assess the suggested models. There is a significant imbalance in the dataset since just 492 of them are fake, which accounts for only 0.172% of the total. Each one has thirty anonymized characteristics and a class label that determines if a transaction is fraudulent. The model's performance was assessed using its accuracy, detection rate, and area under the curve. Each deep learning model AE, CNN, and LSTM was trained and tested under various configurations, including no sampling, SMOTE, ADASYN, and RUS. For hyperparameter tuning, both Random Search and Bayesian Optimization were applied, with each method run four times per model. Python 3.9.7 in the Anaconda 3 environment was used to perform experiments on a Windows 11 (64-bit OS) system with 16 GB RAM and 1 TB SSD.

### 5.1 Hyperparameter Optimization Results

The ideal values for each hyperparameter of three DL models that were improved using random and Bayesian techniques are displayed in Table 1.

**Table 1.** Optimized DL architecture and hyperparameters obtained using both random and Bayesian optimization techniques

| DL | Tuning technique | Sampling techniques | Number of neurons per layer | Batch size | Optimization function | Activation function | Dropout | Loss function | Learning rate |
|---|---|---|---|---|---|---|---|---|---|
| | | Without sampling | 512 | 512 | Adam | Tanh | - | BCE | 0.001 |
| | | RUS | 32 | 32 | Adam | Tanh | - | BCE | 0.001 |
| AE | Random | SMOTE | 512 | 128 | Adam | Relu | - | BFL | 0.001 |

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | ADASYN | 32 | 512 | Adam | Relu | - | BFL | 0.001 |
| | | Without sampling | 256 | 2048 | Adam | Tanh | - | MSE | 0.001 |
| | | RUS | 512 | 128 | Adam | Relu | - | BCE | 0.001 |
| | Bayesian | SMOTE | 512 | 64 | Adam | Relu | - | BFL | 0.001 |
| | | ADASYN | 512 | 64 | Adam | Relu | - | BFL | 0.001 |
| | | Without sampling | 512 | 512 | Adam | Relu | 0.2 | BCE | 0.001 |
| | | RUS | 128 | 32 | Adam | Relu | 0.5 | MSE | 0.001 |
| CNN | Random | SMOTE | 64 | 64 | Adam | Relu | 0.2 | BFL | 0.001 |
| | | ADASYN | 64 | 64 | Adam | Relu | 0.3 | BCE | 0.001 |
| | | Without sampling | 256 | 64 | Adam | Relu | 0.2 | BCE | 0.0001 |
| | | RUS | 256 | 32 | Adam | Relu | 0.3 | BCE | 0.0001 |
| | Bayesian | SMOTE | 128 | 64 | Adam | Relu | 0.5 | BFL | 0.0001 |
| | | ADASYN | 512 | 512 | Adam | Relu | 0.2 | BFL | 0.0001 |
| | | Without sampling | 512 | 64 | Adam | sigmoid | - | BCE | 0.001 |
| | | RUS | 512 | 128 | Adam | sigmoid | - | BFL | 0.001 |
| LSTM | Random | SMOTE | 256 | 128 | Adam | sigmoid | - | BFL | 0.001 |
| | | ADASYN | 64 | 32 | Adam | sigmoid | - | BFL | 0.001 |
| | | Without sampling | 512 | 128 | Adam | sigmoid | - | BCE | 0.001 |
| | | RUS | 512 | 32 | Adam | sigmoid | - | BFL | 0.001 |
| | Bayesian | SMOTE | 512 | 64 | Adam | sigmoid | - | BFL | 0.001 |
| | | ADASYN | 512 | 64 | Adam | sigmoid | - | BCE | 0.001 |

Note: -: represents that a particular hyperparameter is not included in the DL model architecture. BCE: represents Binary Cross Entropy.
BFL: represents Binary Focal Loss. MSE: represents Mean Squared Error

## 5.2. Hyperparameter Search Space and Tuning Insights

Hyperparameter tuning requires defining a search space—the range of values explored to find the best-performing configurations for deep learning (DL) models. This process balances model accuracy with computational cost.

Key hyperparameters and their tested ranges in this study include:

- **Number of neurons**: {32, 64, 128, 256, 512} to control model capacity and prevent overfitting.
- **Dropout rate**: {0.2, 0.3, 0.5} for regularization and improved generalization.
- **Activation functions**: ReLU, sigmoid, and tanh to support non-linear learning.
- **Batch size**: {32, 64, 128, 256, 512, 2048} affecting training efficiency and stability.
- **Optimizers**: SGD, Adagrad, and Adam, with Adam being preferred due to its adaptive learning and momentum handling.

- **Learning rate**: {0.01, 0.001, 0.0001} to control weight update steps.
- **Loss functions**: Binary cross-entropy, focal loss, and MSE, guiding how the model learns.

Across all models, Adam consistently emerged as the best-performing optimizer due to its efficient handling of learning rates and gradient momentum. For the AE model, learning rates remained consistent across tuning methods, suggesting architecture-specific stability. In CNN, activation functions remained consistent through all runs, indicating robustness in tuning outcomes. For LSTM, sigmoid activation performed best, aligning with its internal gating mechanisms and suitability for sequential, time-based fraud patterns.

### 5.3. Performance Evaluation

Table 2 summarizes the results of this section's comparison of the accuracy, detection rate (DR), and area under the curve (AUC) of deep learning models trained using Random Search and Bayesian Optimization across various sampling approaches. Overall, Bayesian Optimization consistently outperforms Random Search, demonstrating its effectiveness in identifying optimal hyperparameter combinations.

**Table 2.** Comparing the accuracy, detection rate, and area under the curve of DL models utilizing random and Bayesian optimization parameters that were trained both with and without sampling. The bolded results represent the best outcomes for all DL models.

| Model | Tuning | Sampling | Acc / DR / AUC |
|---|---|---|---|
| | | None | 95.0 / 90.0 / 92.7 |
| | Random | RUS | 95.1 / 90.4 / 92.7 |
| AE | | SMOTE | 96.1 / 89.7 / 92.9 |
| | | ADASYN | 95.1 / 90.4 / 92.7 |
| | | None | 95.0 / 90.4 / 92.8 |
| | Bayesian | RUS | 95.1 / 89.7 / 92.4 |
| | | SMOTE | 95.1 / 90.4 / 92.8 |
| | | ADASYN | 95.1 / 90.4 / 92.8 |
| | | None | 99.7 / 83.0 / 91.4 |
| | Random | RUS | 99.8 / 87.5 / 93.1 |
| CNN | | SMOTE | 99.8 / 90.7 / 94.7 |
| | | ADASYN | 99.9 / 90.8 / 94.9 |
| | | None | 99.9 / 82.3 / 91.1 |
| | Bayesian | RUS | 99.1 / 86.0 / 92.5 |
| | | SMOTE | 99.7 / 90.4 / 94.7 |
| | | ADASYN | 99.8 / 90.0 / 94.7 |
| | | None | 99.1 / 84.2 / 93.3 |
| | Random | RUS | 99.2 / 87.0 / 94.1 |
| LSTM | | SMOTE | 99.2 / 93.3 / 96.3 |
| | | ADASYN | 99.2 / 92.0 / 95.5 |
| | | None | 99.1 / 83.9 / 93.2 |
| | Bayesian | RUS | 99.2 / 86.9 / 94.0 |
| | | SMOTE | 99.2 / 93.3 / 96.3 |
| | | ADASYN | 99.2 / 92.0 / 95.5 |

Models that incorporate sampling methods perform significantly better than those trained on imbalanced data alone. Among the sampling techniques, SMOTE and ADASYN yield better results than RUS, as they enrich the minority class without discarding valuable majority-class data, improving model generalization. While ADASYN performs best with the AE model due to its focus on difficult-to-learn instances, SMOTE shows better synergy with CNN and LSTM, which benefit from interpolative synthetic samples. Furthermore, CNN and LSTM models with SMOTE sampling outperform AE, primarily because of their ability to model temporal dependencies and extract meaningful features from sequential data. CNN achieves this through layered feature extraction, while LSTM captures long-term dependencies, making both more suitable for fraud detection in time-based transaction sequences. The proposed deep learning models—AE, CNN, and LSTM—demonstrate enhanced discrimination between fraudulent and legitimate transactions when compared to related approaches.

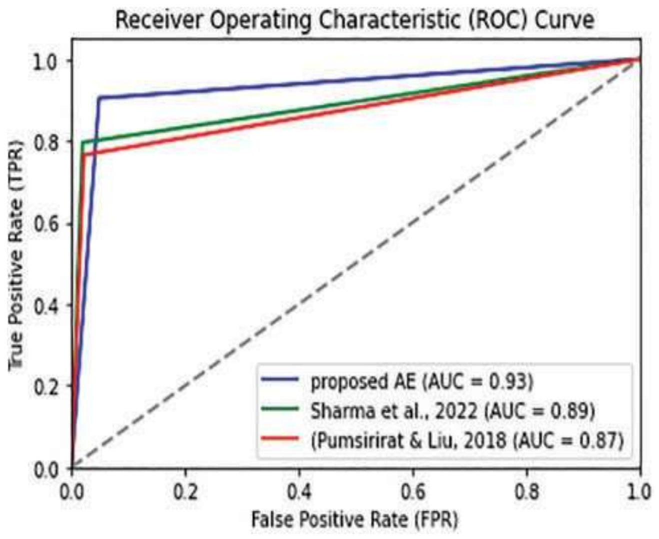

**Fig. 3**. The suggested AE model's ROC curve.

While the LSTM model requires more training time due to its architectural complexity, reduced batch size, and the integration of early stopping, it still outperforms the benchmark model presented in. The suggested models consistently show greater true positive rates (TPR) and lower false positive rates (FPR), which is clearly reflected in the ROC curves as improved performance. This results in The ROC curves nearest the upper-left corner of the plot, indicating robust fraud detection capabilities. Fig 3 shows The suggested AE model's ROC curve.

Fig 4 shows the predicted CNN model's ROC curve compared to that of (Paper 8) and Fig 5 shows the suggested LSTM model's ROC curve compared to that of (Paper 2).

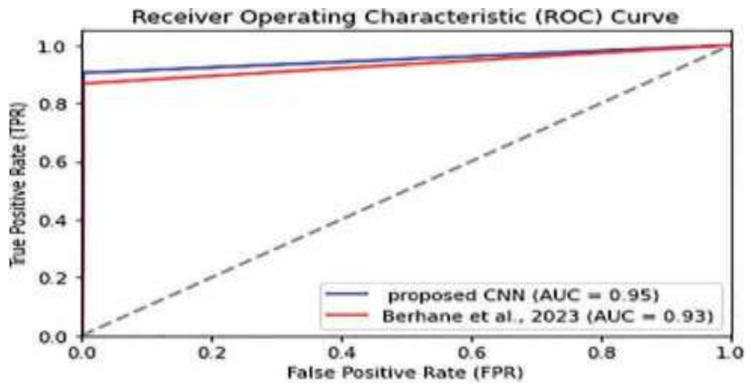

**Fig. 4.** The predicted CNN model's ROC curve compared to that of Berhane et al. [23] (Paper 8).

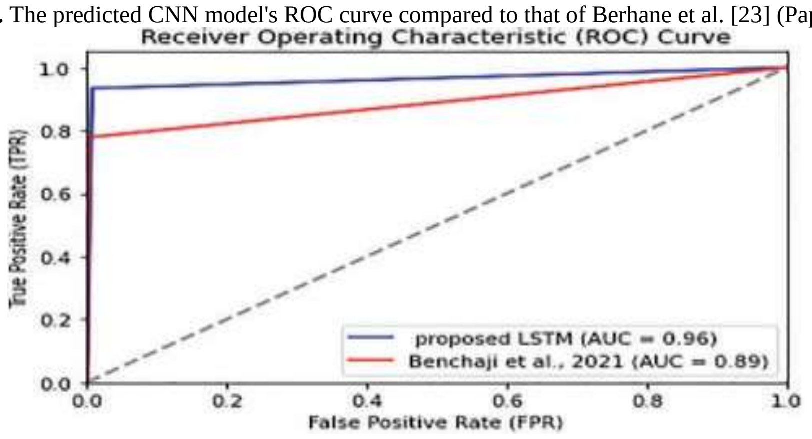

**Fig. 5.** The suggested LSTM model's ROC curve compared to that of Benghazi et al. (Paper 2).

Each model was tested with and without sampling, and under both tuning strategies. Key findings include:

- **Autoencoder:** Achieved ~95.1% accuracy with a detection rate of ~90.4% using SMOTE. Good at anomaly detection but less effective with evolving patterns.
- **CNN:** Performed better than AE, especially with SMOTE and ADASYN. The highest AUC-ROC was 95.1% with Bayesian tuning.
- **LSTM:** Outperformed all other models with a 99.2% accuracy and 93.3% detection rate when trained on SMOTE-balanced data and optimized using Bayesian search.

These results are consistent with those reported by Jurgovsky et al. [3] and Zhou et al. [8], affirming the utility of temporal modeling and attention mechanisms. They also support the efficacy of SMOTE and hyperparameter optimization in imbalanced fraud detection.

## 6 Conclusion

Here, we used state-of-the-art optimization and sampling methods to assess deep learning models' ability to identify credit card fraud. We demonstrated that LSTM, when trained on

SMOTE-enhanced data and optimized via Bayesian search, achieves superior performance in detecting fraudulent transactions in imbalanced datasets. Our findings align with recent literature [1]– [3], [8], reinforcing the importance of combining data preprocessing, architecture selection, and hyperparameter tuning. Future research may explore hybrid ensemble models, real-time detection pipelines, and privacy-preserving training methods such as federated learning

## References

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015.

[2] U. Fiore, A. De Santis, F. Perla, P. Zanetti, and F. Palmieri, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Information Sciences*, vol. 479, pp. 448–455, Apr. 2019.

[3] J. Jurgovsky et al., "Sequence classification for credit-card fraud detection," *Expert Systems with Applications*, vol. 100, pp. 234–245, Jun. 2018.

[4] A. Dal Pozzolo, O. Caelen, Y. A. Le Borgne, S. Waterschoot, and G. Bontempi, "Calibrating probability with undersampling for unbalanced classification," in *Proc. IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, Cape Town, South Africa, 2015, pp. 159–166.

[5] C. Bahnsen, D. Aouada, A. Stojanovic, and B. Ottersten, "Feature engineering strategies for credit card fraud detection," *Expert Systems with Applications*, vol. 51, pp. 134–142, Jun. 2016.

[6] Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bontempi, "Credit card fraud detection: A realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, Aug. 2018.

[7] J. West and M. Bhattacharya, "Intelligent financial fraud detection: A comprehensive review," *Computers & Security*, vol. 57, pp. 47–66, Apr. 2016.

[8] Y. Zhou, H. Han, and Y. Zhang, "Credit card fraud detection using CNN and attention mechanism," in *Proc. Int. Conf. on Artificial Intelligence*, Chengdu, China, 2019, pp. 1–5.

[9] F. Carcillo et al., "Scarff: A scalable framework for streaming credit card fraud detection with Spark," *Information Fusion*, vol. 41, pp. 182–194, May 2018.

[10] Phua, V. Lee, K. Smith, and R. Gayler, "A comprehensive survey of data mining-based fraud detection research," *arXiv preprint arXiv:1009.6119*, Sep. 2010. [Online]. Available: https://arxiv.org/abs/1009.6119

[11] E. W. T. Ngai, Y. Hu, Y. H. Wong, Y. Chen, and X. Sun, "The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature," *Decision Support Systems*, vol. 50, no. 3, pp. 559–569, Feb. 2011.

[12] Tanyu, W., Huber, S., Kutyniok, G., & Petersen, P. (2022). Deep learning methods for partial differential equations and related parameter identification problems: A survey. GAMM-Mitteilungen, 45(3), e202200008. Wiley. DOI: 10.1002/gamm.202200008

[13] Pironneau, O. (2021). Supervised learning and applied mathematics. In: Handbook of Numerical Analysis, Vol. 23, pp. 39–70. Elsevier. DOI: 10.1016/bs.hna.2021.04.002

[14] Meenu. (2024). The role of mathematics in artificial intelligence and machine learning. In: Proceedings of International Conference on Emerging Technologies in Computer Engineering (ICETCE 2024), pp. 221–231. Springer, Singapore.

[15] Jin, S., Wu, X., & Zhang, L. (2024). Research and application implementation of deep learning algorithms based on Python. In: Proceedings of International Conference on Computational Intelligence and Intelligent Systems (CIIS 2024), Lecture Notes in Networks and Systems, vol. 958, pp. 189–199. Springer, Singapore. DOI: 10.1007/978-981-97-4121-2_14