Deep Learning Methods For Identifying Credit Card Fraud Using Advanced Techniques To Increase Security

Sachin Kumar Soni[1], **Dr. Balveer Singh**[2]

Department of Computer Science[1,2], P.K. University, Shivpuri, M.P. India.

sachinkumarsoni185@gmail.com, adm.pkit@gmail.com

Abstract- Applying the Paysim synthetic dataset, this project aims to develop and evaluate ML models for mobile money fraud detection. We aim to evaluate how well sophisticated approaches, especially ANN and hybrid architectures, can detect fraudulent actions. There are multiple essential stages to the approach. First, we gather the Paysim dataset, which includes information like the kinds of transactions, their prices, account balances, and any signs of fraud. Exploratory Data Analysis (EDA) gives insights into transaction patterns after data pretreatment cleans and standardizes the dataset. It is feasible to build or train several models using the best optimization strategy after data is partitioned into testing and training sets. Hybrid designs (including RNNs, LSTMs, and GRUs) and artificial neural networks comprise these models. Examining the model's accuracy or predictive capacities allows us to gauge its performance. Among the models that were constructed, the ANN model stands out to be the top performer due to its exceptional accuracy and great predictive abilities. Its ability to efficiently capture complicated fraud patterns is due to its sequential architecture, which is complemented by many hidden layers and optimization approaches. The ANN's performance highlights the significance of using sophisticated deep learning architectures to fraud detection jobs. This study provides useful insights for financial organizations looking to improve their fraud detection mechanisms by highlighting the necessity of employing realistic synthetic datasets to system development and testing.

Keywords- *Artificial Neural Networks, Deep Learning, Paysim Dataset, and Fraud Detection*

1. Introduction:

Both customers and financial institutions face significant operational challenges and financial risks due to the proliferation of credit card theft in the modern digital age. The sophistication of fraudulent activity is growing in tandem with the exponential growth of online transactions.

Traditional methods that rely on rules to detect credit card fraud often fall short because they can't adapt to new and different types of fraud [1]–[6]. Consequently, there has been an effort to go deeper into more advanced approaches, particularly within the domains of AI and ML.

Deep learning is a machine learning approach that attempts to mimic the brain's information processing capabilities; it has the potential to significantly decrease credit card fraud. Models based on deep learning, namely those that use multi-layer neural networks, have revolutionized fraud detection. Automated feature extraction and learning from large datasets is where these models really shine, allowing them to spot suspicious trends and outliers that may indicate fraud. [7]–[13]. Without using pre-defined criteria or manual feature extraction, deep learning systems can scrutinize raw transaction data for concealed correlations and nuanced signs suggesting fraud. Due to its ability to learn and adapt, deep learning is a formidable tool in the fight against credit card fraud.

Figure 1 Credit card fraud detection

When implementing deep learning for the purpose of detecting credit card fraud, many critical processes are typically involved[14]–[20]. Acquiring a massive and varied dataset for transaction records, including both genuine and fraudulent ones, is the first order of business. Following this, the dataset undergoes pre-processing to correct missing values, normalize features, and alter categorical variables. Just after data preprocessing, the first step in building and testing the model is to partition the data into sets of validation, testing, and training. Step two involves constructing and training a complex neural network to distinguish between legitimate and fraudulent transactions using patterns in the input attributes, using the training data that has been provided. Adjusting the model's hyper parameters and using methods like cross-validation boost its performance[21]–[27]. This helps avoid overfitting and guarantees the model's ability to generalize to novel, unobserved data.

Deep learning's ability to efficiently handle massive amounts of data and spot intricate patterns that less advanced models might miss is a major plus when it comes to fraud detection. Long short-term memory networks (LSTMs) and recurrent neural networks (RNNs) do very well when assessing data that consists of consecutive transactions. This is because they are able to detect trends and temporal linkages that static models could miss. Analyzing transaction data has also shown to be a successful use of convolutional neural networks (CNNs), which are often associated with picture processing. Using this method, the convolutional neural network (CNN) finds spatial correlations between different features by treating each transaction record as a feature map [28]–[32].

Despite deep learning's impressive advancements, there are still several issues and concerns that require addressing. The issue of interpretability is important to serious concern. Many people think deep learning models are "black boxes" due to the complicated and hard to understand ways they make decisions. In the financial sector, where understanding the logic behind a fraud detection decision is crucial for regulatory compliance and client confidence maintenance, the lack of transparency can be a major obstacle. In order to address this issue, scientists are investigating model-agnostic interpretability methods. The goal of these techniques is to improve the model's performance without sacrificing the accuracy of its predictions. When it comes to combating credit card fraud, deep learning is head and shoulders above the competition thanks to its superior accuracy and versatility in detecting fraudulent transactions. Using deep neural networks, which can learn from large amounts of transaction data, financial organizations can strengthen their fraud prevention. Research and development efforts in this area are continuously improving the efficiency and dependability of deep learning models, even while challenges like interpretability and class imbalance exist. The development of more secure and effective financial systems depends on this advancement [33]–[37].

2. Literature review:

[38] Shown efficacy when encountering two obstacles: an uneven dataset and the development of fraudulent behaviors. Introducing the auto encoder for the purpose of evaluating unsupervised deep learning; data balance is crucial. Training three models was done using a dataset that contained 284,807 trades. As part of the pre-processing, features were selected, data was balanced, and normalization was performed. Model parameters were fine-tuned and enhanced after the installation. Evaluation using the area under the curve (AUC), recall, accuracy, precision, or confusion matrix. For legitimate transactions, the chosen model earned a f1-score of 93%.

[1] Feature reduction and investigated data sampling approaches for detecting credit card fraud. Employed PCA and CAE for feature extraction. In order to gather data, we tested SMOTE, RUS, and SMOTE Tomek. Tested a number of classifiers, such as Light GBM, Random Forest, CatBoost, or XGBoost. The most effective method was RUS followed by CAE.

[39] There was a corresponding spike in credit card fraud as online shopping became more popular. Valuing the identification for fraud for. The security of online payment. Combining DL using oppositional cat swarm optimization, the OCSODL-CCFD approach selects features. I tuned the BiGRU model using CKHA. Results from simulations show that the OCSODL-CCFD model outperforms its competitors.

[39] Fraud rises alongside e-commerce growth and online payments' convenience. ML increasingly deployed for fraud detection. Studied ML methods for CCFD and data confidentiality, proposing mixed approach employing ANNs in federated learning. Enhanced CCFD accuracy while safeguarding privacy noted.

[40] Increased credit card fraud prompts exploration of ML models for detection. Model building is made easier with Just-Add-Data (JAD) since it automates evaluation, hyper parameter tuning, and method selection. User-friendly interface enables easy application, display, and sharing of results within credit card organizations. JAD-selected model identifies 32 out of 39 fraudulent transactions, demonstrating promising predictive performance comparable to existing approaches[40].

Table 1 Literature summary

Author / Year	Method	Research gap	Controversies	References
Saxena/ 2022	Automated Detection of Credit Card Fraud	Bringing up under-discussed strategies for identifying instances of credit card theft .	Moral questions regarding the appropriateness of data collection and use in the fight against credit card fraud.	[41]
Alharbi/2022	Using innovative deep learning techniques to better detect credit card fraud.	Poor methods for detecting fraud due to a lack of attention to class imbalance.	Concerns about data privacy in fraud detection give rise to ethical difficulties.	[8]
Moumeni/2022	A look at the strengths and weaknesses of many ML systems developed to combat credit card fraud.	Investigation on new methods of e-commerce fraud detection has been minimal.	Ethical concerns arise regarding data usage in fraud detection methods.	[10]
Ashraf/ 2022	Research on the	Little studies	Ethical debates	[9]

	identification of credit card fraud using ML and DL approaches compared.	have compared the effectiveness of ML and DL techniques.	arise over privacy concerns in fraud detection methods.	
Zhang/2022	Utilization for anomaly detection techniques has enhanced the detection of credit card fraud.	Anomaly identification for credit fraud datasets with imbalances has not been extensively studied.	Methods for detecting fraud spark debates regarding their efficacy and ethics.	[6]

3. Research Methodology

There are several critical stages to follow when developing and testing deep learning models to identify fraudulent transactions using the Paysim synthetic dataset. The first stage in creating a practical model for mobile money transactions is gathering the dataset. It contains information like the kind of transaction, its value, the account balance, and any signs of fraud. The next step is data pre-processing, which includes checking for missing values and standardizing numerical properties to ensure consistency. Exploratory data analysis (EDA) employs various visualizations to better comprehend the data by identifying patterns or behaviors in transactions. Afterwards, we split the data in half, 80/20, so we have one set for training & another for testing, so we can see how well the model does on fresh data. An Artificial Neural Network (ANN) or hybrid model may have several hidden layers constructed using the ReLU activation function and a sigmoid function for binary classification (LSTM, RNN, GRU). With 20% set aside for validation, the models are trained over 32 batches and 50 epochs using the binary cross-entropy loss function or the Adam optimizer. Deep learning is useful for detecting complicated patterns of fraud, and this all-encompassing method guarantees excellent accuracy and strong prediction abilities.

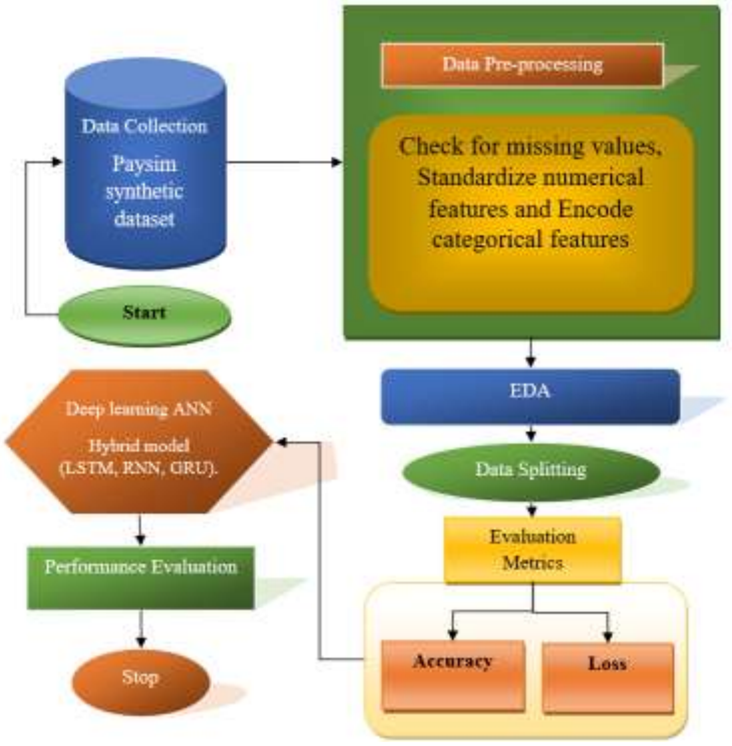
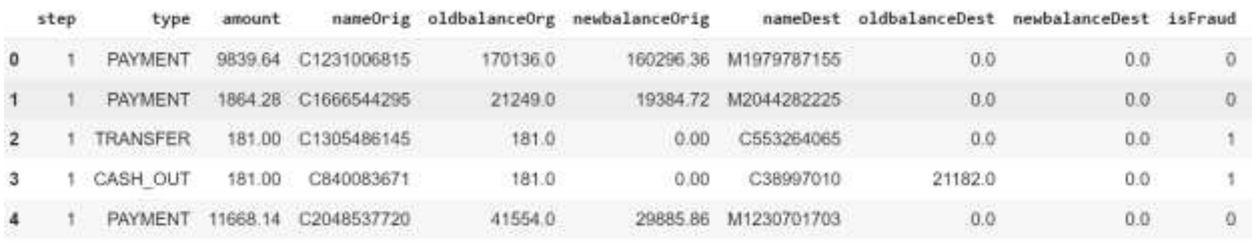

Data Collection
Paysim synthetic dataset
Start
Data Pre-processing
Check for missing values, Standardize numerical features and Encode categorical features
EDA
Data Splitting
Evaluation Metrics
Accuracy
Loss
Deep learning ANN Hybrid model (LSTM, RNN, GRU).
Performance Evaluation
Stop

Figure 2 Proposed Flowchart

3.1 Data Collection

An enormous dataset developed for the express purpose of detecting fraudulent mobile money transactions is the Paysim synthetic dataset. This dataset replicates actual financial transactions occurring at regular hourly intervals. Information such as the kind of transaction (e.g., PAYMENT or TRANSFER), its value, the account numbers of the sender and receiver, the balances of the accounts before and after the transaction, or signs of possible fraud are all part of each transaction record. Aside from the obvious "isFlaggedFraud" for suspicious transactions, the dataset also contains the crucial "isFraud" to determine whether a transaction was fraudulent. This comprehensive and meticulous dataset is extremely important for the development, testing, and validation of machine learning models that are designed to detect financial fraud in mobile money systems. It is an essential resource for researchers and practitioners seeking to enhance fraud detection systems, providing a realistic simulation of transaction patterns and potential fraud scenarios. The dataset can be accessed and downloaded from this link: https://www.kaggle.com/datasets/ealaxi/paysim1

This dataset offers a comprehensive framework for comprehending transaction behaviours and constructing advanced fraud detection systems, rendering it a crucial resource for financial data scientists and analysts.



	step	type	amount	nameOrig	oldbalanceOrg	newbalanceOrig	nameDest	oldbalanceDest	newbalanceDest	isFraud
0	1	PAYMENT	9839.64	C1231006815	170136.0	160296.36	M1979787155	0.0	0.0	0
1	1	PAYMENT	1864.28	C1666544295	21249.0	19384.72	M2044282225	0.0	0.0	0
2	1	TRANSFER	181.00	C1305486145	181.0	0.00	C553264065	0.0	0.0	1
3	1	CASH_OUT	181.00	C840083671	181.0	0.00	C38997010	21182.0	0.0	1
4	1	PAYMENT	11668.14	C2048537720	41554.0	29885.86	M1230701703	0.0	0.0	0

Figure 3 Preview of Dataset

3.2 Data Pre-processing

Prior to analysis and model training, the data from the Paysim synthetic dataset undergoes a series of steps known as data preparation. To begin, we ensure that the dataset is completely inclusive by using the df.isnull().sum() function to find any missing values. Standardising the numeric features in the dataset is crucial to ensure uniformity, as it consists of different sorts of transactions and monetary values. With this job, we will isolate the DataFrame columns that hold numerical values, namely the float64 or int64 columns. For numerical column standardization, it uses the sklearn.preprocessing module's Standard Scaler function. By first running the scaler over the whole dataset, we can get the average and standard deviation of each feature. Next, we utilised these numbers to make the necessary adjustments to the dataset. Now, every feature has a standard deviation of one and an average of zero. Standardisation is essential because it guarantees that all features have an equal contribution to the model training process. This prevents features with greater numerical ranges from overpowering the learning algorithm. We return the data to its original Data Frame format after the developers have scaled it. The original column names are preserved in this way to make sure everything is easy to understand. Next, you can divide the preprocessed DataFrame into sets of training and testing data or machine learning tasks, or do further analysis on it.

This pseudocode outlines the steps required to preprocess the dataset
1. Check for missing values - missing_values = df.isnull().sum() - print(missing_values) 2. Select numeric columns - numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns 3. Initialize scaler - scaler = StandardScaler() 4. Fit scaler on numeric columns - scaler.fit(df[numeric_cols]) 5. Transform numeric columns - df_scaled_values = scaler.transform(df[numeric_cols]) 6. Convert scaled data to DataFrame - df_scaled = pd.DataFrame(df_scaled_values, columns=numeric_cols) 7. Print scaled DataFrame - print(df_scaled)

3.3 Exploratory Data Analysis (EDA)

By illuminating common patterns and behaviors of transactions, the offered exploratory data analysis (EDA) helps to better understand financial data trends. Based on the line graph, it appears that there have been more minor transactions than large ones. Payments, transfers, and debits are examples of cashless transactions, and a bar graph can demonstrate how prevalent these are compared to traditional cash transactions. Among these cashless transactions, payments stand out. A boxplot shows that compared to cash transactions, the monetary values of payment, transfer, and debit transactions tend to be higher. This suggests that non-cash transactions often include larger sums of money. According to a scatter plot, there is a positive correlation between the original & current account balances, suggesting that higher initial balances are more likely to be maintained over time. These insights are vital for financial institutions and analysts as they aid in spotting transaction patterns, comprehending consumer behaviour, and making well-informed decisions about risk management, fraud detection, and financial planning. Through the examination of these patterns, organisations can more effectively meet customer demands, streamline transaction procedures, and improve overall financial robustness.

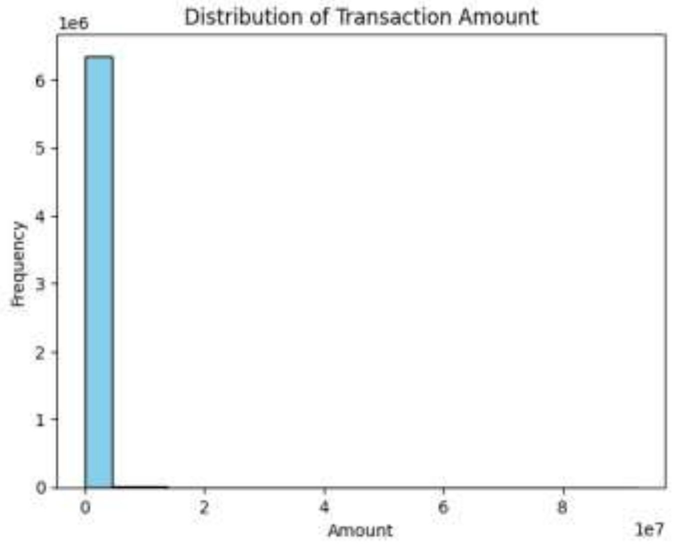


Figure 4 Distribution of Transaction Amount

The figure shows a graph with a straight line representing the distribution for transaction amounts. On one side, we can see the overall number of transactions, and on the other, we can see the frequency with which those transactions occurred. There will be more transactions of a given value when the frequency rises. For instance, the graph shows that small-value transactions occur frequently (on the left side of the x-axis) while large-value transactions occur far less frequently (on the right side of the x-axis).This graph does not provide enough information to determine the exact type of transactions or the money used

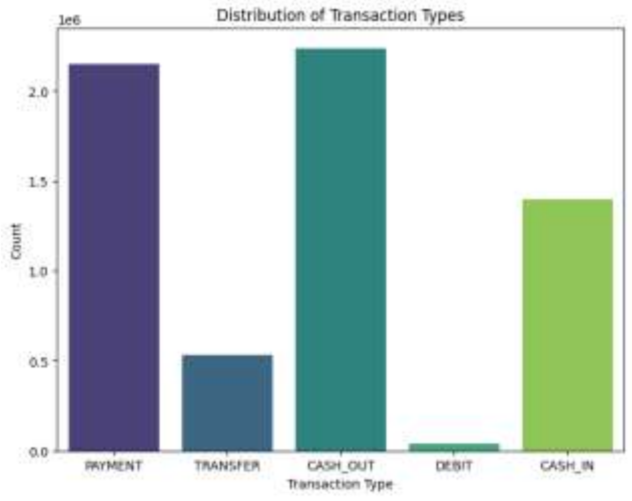


Figure 5 Count Plot of Distribution of Transaction Types

The visual representation is a bar graph illustrating the quantity of transactions across several categories. The total number of transactions is shown on the y-axis, while the x-axis represents the various types of transactions. As an illustration, it seems that there are a greater number of "payment" transactions compared to "cash_out" transactions. In comparison to cash transactions like cash outs and cash in, it seems like there are a lot more cashless transactions like payments, transfers, and debits.

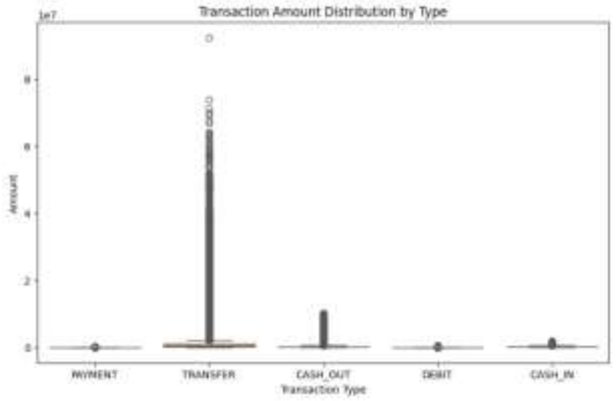


Figure 6 Boxplot of Transaction Amount Distribution by Type

The boxplot in Figure 6 illustrates the impact of various transaction types on the distribution for amounts. The middle 50% of transactions are depicted in the interquartile range in the box graphic. Centered within the box is the median amount. Whiskers cover a distance 1.5 times the IQR, or the interval between the two extremes of a data set. The relative heights of the boxes suggest that more money is coming in than going out. In general, the transaction amounts for payments, transfers, and debits appear to be more than those for cash in and cash out transactions.

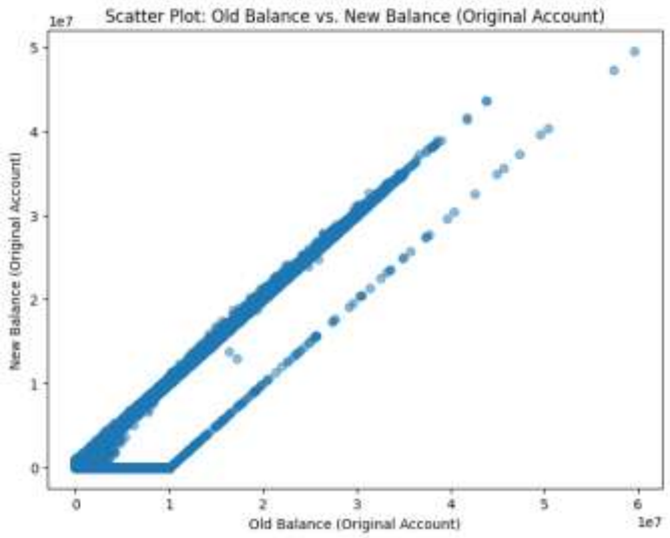


Figure 7 Scatter plot of Old balance(Original Account)

The graphic depicts a scatter plot that compares the original account's previous balance to its new balance. The location of each data point on the graph represents the relationship between the prior and current balance of a certain account, and each data point is associated with a different account. There exists a positive association, indicating that accounts with greater previous balances tend to exhibit higher current balances as well.

3.4 Data Splitting

Data splitting is an essential part of machine learning for testing a model's performance on new data. It is possible to use the 'train test split' function in the'sklearn.model_selection' package to partition our dataset into a training set and a testing set. We start by deleting the 'isFraud' and 'isFlaggedFraud' columns within the DataFrame 'df' in order to make room for our features (X). Within these columns, you will discover the goal variable along with some data that may not be predictive. The goal variable (y) is represented by the 'isFraud' column in this case. The proper proportions for training and testing our model can be achieved by halving the data: 80% for training or 20% for testing. When training, this is achieved using the 'train_test_split(X, y, test_size=0.2, random_state=42)' function. The function call generates all four datasets: 'X_train' and 'y_train' for model training, and 'X_test' or 'y_test' for model testing. Setting a random state ensures that the split will occur again. If you want to develop a reliable prediction model, you must ensure that you have divided the data correctly. Using this code is all it takes to get the dataset shapes out: In addition to the coordinates, this function will also return the data forms of the training (X_train) and test (X_test) sets. This is only possible with massive training datasets as opposed to little test sets.

Pseudocode for data splitting

```
1. Import necessary libraries:
   Import train_test_split function from appropriate library/module.

2. Load the dataset:
    Load the dataset into a suitable data structure (e.g., DataFrame if using pandas in Python).

3. Define features (X) and target variable (y):
```

Select feature columns (all columns except the target and any non-predictive columns).

Identify and select the target column.

4. Create separate sets of data for testing and training:

Split X and y into sets for training and testing using the train_test_split function or method. Give an example of how much data will be used for testing: 20% for testing and 80% for training.

For consistency, you can choose to use a random state or seed.

5. Verify the split:

Print or inspect the shapes of resulting training and test sets to confirm the correctness of the split.

Pseudo code example:

4.2 Modeling

Finding credit card fraud with deep learning requires ANN models to be integrated with hybrid architectures like LSTM, RNN, and GRU. But nonetheless. There are fifty, forty, thirty, twenty, or ten neurons in each of the five hidden layers that make up the ANN model's sequential structure.

As a first step, the method uses standard preprocessing techniques to scale features and partition data. For binary classification, the output layer employs a sigmoid function, whereas the activation function for each layer is the ReLU. A binary cross-entropy loss function and the Adam optimizer are used by the model to guarantee convergence and successful learning. Iterative updates are used to alter the model's weights in each one of the fifty epochs that the comprise the training phase. While training, we use a batch size of 32 and set aside 20% for the data for validation. The amazing accuracy is the consequence of evaluating the training history and model performance with test data. This approach exemplifies how deep learning can detect intricate patterns of fraudulent transactions, surpassing traditional methods and providing superior prediction abilities while actively adapting to evolving fraud strategies.

Pseudo Code of Model Implemetation

```
 Import necessary libraries
import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM, SimpleRNN, GRU
from tensorflow.keras.initializers import glorot_uniform
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

Load dataset
X, y = load_data()

 Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

 Feature scaling
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

 Define the number of hidden layers and neurons in each layer for ANN
hl = 5
nohl = [50, 40, 30, 20, 10]

 Create the Sequential model for ANN
classifier = Sequential()
```

```
 Add hidden layers to ANN model
for i in range(hl):
   if i == 0:
      classifier.add(Dense(nohl[i],  input_dim=X_train.shape[1],  kernel_initializer='uniform',
activation='relu'))
   else:
      classifier.add(Dense(nohl[i],                    kernel_initializer=glorot_uniform(seed=0),
activation='relu'))

 Add output layer to ANN model
classifier.add(Dense(1, kernel_initializer=glorot_uniform(seed=0), activation='sigmoid'))

 Compile the ANN model
classifier.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])

 Train the ANN model
history = classifier.fit(X_train, y_train, epochs=50, batch_size=32, validation_split=0.2)

 Evaluate the ANN model on test data
test_loss, test_accuracy = classifier.evaluate(X_test, y_test)
print("Test Accuracy:", test_accuracy)

 Create hybrid model with LSTM
hybrid_model_lstm = Sequential()
 Add LSTM layers as required
 Add Dense layers as needed
 Compile and train the hybrid LSTM model

 Create hybrid model with Simple RNN
```

```
hybrid_model_rnn = Sequential()
 Add Simple RNN layers as required
 Add Dense layers as needed
 Compile and train the hybrid Simple RNN model

 Create hybrid model with GRU
hybrid_model_gru = Sequential()
 Add GRU layers as required
 Add Dense layers as needed
 Compile and train the hybrid GRU model

 Evaluate all hybrid models on test data and compare results
 Evaluate hybrid LSTM model
 Evaluate hybrid Simple RNN model
 Evaluate hybrid GRU model
 Compare their test accuracies and other relevant metrics
```

Table 2 Hyperparameter Details

Number of Hidden Layers	[3, 4, 5, 6, 7]
Neurons per Layer	[10, 20, 30, 40, 50]
Activation Function	['relu', 'tanh', 'sigmoid']
Optimizer	['adam', 'sgd', 'rmsprop']
Batch Size	[16, 32, 64, 128]
Epochs	[50, 100]

1) Adam Optimization

A hybrid method, Adam improves performance by combining momentum or RMSprop optimization techniques. The Adam optimization method's parameters Θ are used in the following computation to derive the update rule:

$$\theta_{t+1} = \theta_t \frac{\alpha \cdot m_t}{\sqrt{v_t} + \epsilon} \tag{1}$$

Alpha represents the learning rate, theta stands for the model parameters, m t for the exponentially decaying average of previous gradients, v t for the exponentially decaying average of previous squared gradients, and t for the small constant epsilon, which maintains the division by zero method of the Adam optimization update.

2) ReLU

One common activation function within neural networks is the rectified linear unit, or ReLU. If the input is affirmative, it will directly report the current value; otherwise, it will output zero. To address the problem of training-related vanishing gradients, ReLU encourages activation sparsity. It is often used in deep learning architectures because of its simplicity and effectiveness.

$$f(x) = \max(0, x) \tag{2}$$

In this case, x stands for the function's input, and f(x) = f(x) is the result. When an input is positive (x), the ReLU function returns it directly; when an input is negative, it returns zero. This simple piecewise-linear function is the main training function for neural networks because it promotes sparsity and reduces the vanishing gradient issue.

3) Sigmoid

The input values are converted to a binary range of zeros and ones utilising the sigmoid activation function. Incorporating its smooth gradient into neural networks is a popular practice, and it works wonderfully for binary classification issues.

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{3}$$

The input value is dx, the base of the natural logarithm is e, and the output of the sigmoid function is (x). Any input can be changed by this function into a number between 0 and 1.

4. Result & Discussion

4.1 Performance Evaluation

The proposed models' effectiveness was evaluated using accuracy and loss as performance indicators. We evaluated the model's capacity to accurately identify emotions. It showed the proportion of instances where the predictions were correct in respect to the total number of incidents. At the same time, the loss metric was used to measure the gap between the model's predictions and the actual values, which allowed us to assess the model's training success and error-correction ability. Considering accuracy and loss, we put the models through their paces to assess their ability to accurately identify emotions and enhance prediction outcomes.

4.1.1 Accuracy

accuracy is the primary criteria used to assess deep learning classification models. The metric yields a numerical measure for prediction accuracy by calculating the fraction of properly predicted occurrences compared to the total number for instances in a given dataset. Dividing the total amount for forecasts by the proportion of accurate forecasts is the mathematical way to find a model's accuracy. While accuracy is a good metric to use to evaluate a model's performance, it might not be the best choice when one class an outlier in a dataset. The accuracy metric is widely used in various applications due to its straightforward computation and critical insights it provides on classification algorithm performance.

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)} \tag{4}$$

4.1.2 Loss

Due to its ability to quantify the discrepancy between expected and actual values, the loss metric—also referred to as the cost or objective function—is an essential tool in deep learning. It functions as a statistic for assessing how well the model performed during training. Reducing the loss function is the primary objective of deep learning model training, which raises the model's predicted accuracy. Regression, classification, and reinforcement learning are just a few of the many task-specific loss functions available.

$$Loss = -\frac{1}{m}\sum_{i=1}^{m} \mathcal{Y}i.\log(\mathcal{Y}i) \tag{5}$$

Table 3 Performance Evaluation of Proposed Models

Model	Loss	Accuracy
Artificial Neural Network	7.4	100
Hybrid LSTM,RNN and GRU	6.8	52.08

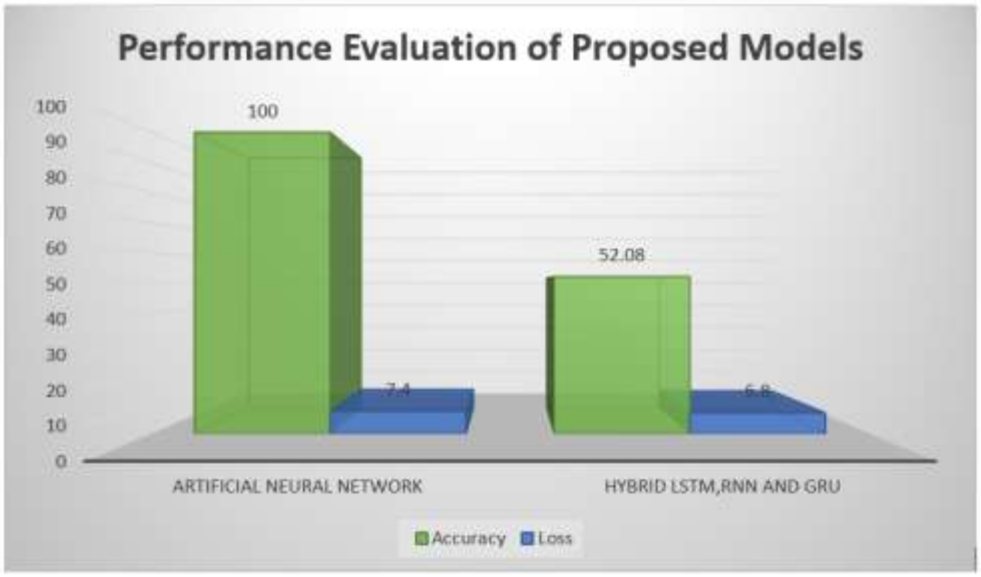


Figure 8 Performance Evaluation Graph

In Table 3 and Figure 8, we can see the results of the performance evaluations of the two models that were suggested: an ANN and a hybrid model that combined LSTM, RNN, and GRU techniques. Accuracy and loss are the evaluation criteria that are utilized. With a loss of 7.4, the ANN-trained model reaches a flawless accuracy score of 100. Since the model yields perfect results in the evaluation dataset, it follows that its predictions are quite accurate. The low loss value further confirms the model's effectiveness in minimising the discrepancy between the anticipated outputs and the actual target values.

Table 4 Comparative Analysis between Previous work and Proposed Work

Models	Accuracy	References
Deep learning Approach	96.33	[21]
LSTM	98.17	[14]
Proposed Artificial Neural Network	100	--

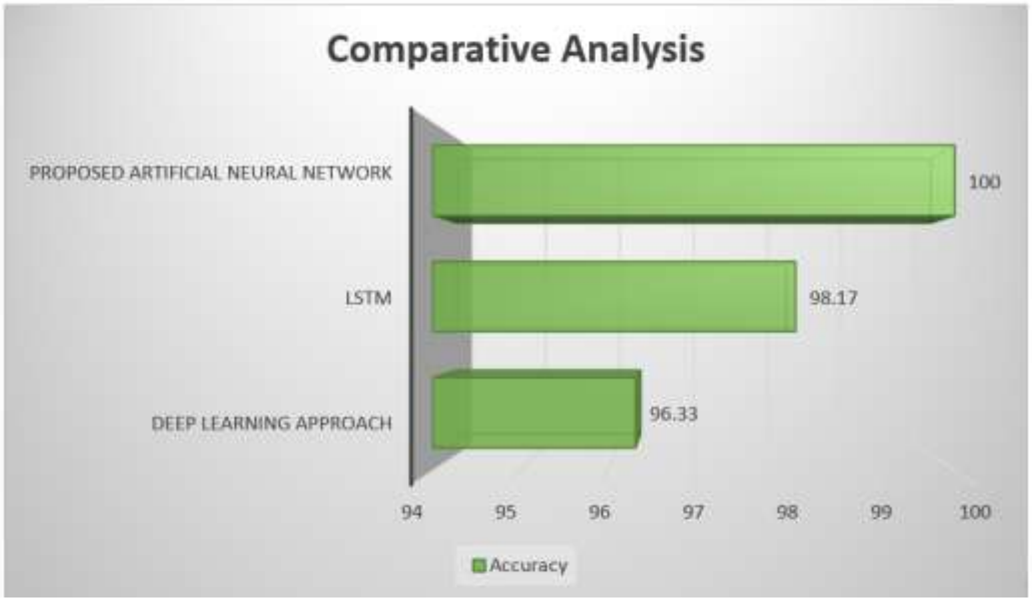


Figure 9 Comparative Analysis Graph

Table 4 & Figure 9 provide a comparison of the proposed ANN model with previous studies, with an emphasis on accuracy metrics. We evaluate the suggested ANN with an LSTM model and a Deep Learning strategy. With the Deep Learning Approach, we were able to attain an accuracy of 96.33%. This suggests that there is a small window of opportunity for enhancement, despite the approach's robustness. As a whole, deep learning (which encompasses several models and techniques) does a good job of solving complicated problems because it learns hierarchical representations. With a 98.17% accuracy rate, the LSTM model is the best of the bunch. When context is more essential than sequences, LSTMs are an excellent choice because they can analyse sequential input and record long-term dependencies. When contrasted with the more generalised deep learning method, its superior accuracy demonstrates how well it handles specific sequential patterns within the dataset. Complete perfection is achieved by the proposed

ANN model. The ANN model has successfully classified all incidents in the test dataset, as demonstrated by this stunning outcome. The ANN may have done a better job of learning the dataset than the LSTM model, as it achieved 100% accuracy. This might be due to the fact that the ANN is very effective for the given task due to its one-of-a-kind design, training methods, or hyper parameters. By comparing the proposed ANN model to previous techniques, we can see how much better it performs. The ANN model's 100% accuracy indicates an extraordinary fit for the given problem, demonstrating breakthroughs in model architecture and training approaches. Deep learning and LSTM models are robust and effective, too.

5. Conclusion

With this all-encompassing approach, we can show that state-of-the-art Deep learning systems can detect mobile money fraud using the Paysim fake dataset. Ensured the dataset was error-and inconsistency-free and uniformly prepared for model training by undertaking comprehensive data collecting and preprocessing. Modelling was based on critical insights about transaction patterns that emerged from Exploratory Data Analysis (EDA). The most promising of the constructed models was the Artificial Neural Network (ANN), which demonstrated strong predictive capacity and a high degree of accuracy. The model's sequential design, numerous hidden layers, and effective optimization techniques allowed it to accomplish outstanding results. The artificial neural networks (ANN) ability to learn and converge, enabled by the rectified linear unit (ReLU) activation function with the Adam optimizer, proved its superiority in detecting complex fraud patterns. Because ANNs are so effective, using complicated deep learning structures is essential for jobs that involve fraud detection. Financial institutions looking to enhance their fraud detection systems can greatly benefit from this product due to its high level of accuracy and predictive power. In order to build and test fraud detection algorithms, this work stresses the significance of using real synthetic datasets like Paysim. By setting a precedent for future studies in financial data analytics, the methodology improves fraud detection and shows how powerful machine learning can be when applied to real-world problems.

References

[1] Z. Salekshahrezaee, J. L. Leevy, and T. M. Khoshgoftaar, "The effect of feature extraction and data sampling on credit card fraud detection," *J. Big Data*, vol. 10, no. 1, 2023, doi: 10.1186/s40537-023-00684-w.

[2] Z. Faraji, "A Review of Machine Learning Applications for Credit Card Fraud Detection with A Case study," *SEISENSE J. Manag.*, vol. 5, no. 1, pp. 49–59, 2022, doi: 10.33215/sjom.v5i1.770.

[3] G. Sasikala *et al.*, "An Innovative Sensing Machine Learning Technique to Detect Credit Card Frauds in Wireless Communications," *Wirel. Commun. Mob. Comput.*, vol. 2022, no. i, 2022, doi: 10.1155/2022/2439205.

[4] A. Singh, A. Jain, and S. E. Biable, "Financial Fraud Detection Approach Based on Firefly Optimization Algorithm and Support Vector Machine," *Appl. Comput. Intell. Soft Comput.*, vol. 2022, no. Cc, 2022, doi: 10.1155/2022/1468015.

[5] H. Xu, G. Fan, and Y. Song, "Application Analysis of the Machine Learning Fusion Model in Building a Financial Fraud Prediction Model," *Secur. Commun. Networks*, vol. 2022, 2022, doi: 10.1155/2022/8402329.

[6] Y. F. Zhang, H. L. Lu, H. F. Lin, X. C. Qiao, and H. Zheng, "The Optimized Anomaly Detection Models Based on an Approach of Dealing with Imbalanced Dataset for Credit Card Fraud Detection," *Mob. Inf. Syst.*, vol. 2022, 2022, doi: 10.1155/2022/8027903.

[7] V. Plakandaras, P. Gogas, T. Papadimitriou, and I. Tsamardinos, "Credit Card Fraud Detection with Automated Machine Learning Systems," *Appl. Artif. Intell.*, vol. 36, no. 1, 2022, doi: 10.1080/08839514.2022.2086354.

[8] A. Alharbi *et al.*, "A Novel text2IMG Mechanism of Credit Card Fraud Detection: A Deep Learning Approach," *Electron.*, vol. 11, no. 5, pp. 1–18, 2022, doi: 10.3390/electronics11050756.

[9] M. Ashraf, M. A. Abourezka, and F. A. Maghraby, "A Comparative Analysis of Credit Card Fraud Detection Using Machine Learning and Deep Learning Techniques," *Lect. Notes Networks Syst.*, vol. 224, pp. 267–282, 2022, doi: 10.1007/978-981-16-2275-5_16.

[10] L. Moumeni, M. Saber, I. Slimani, I. Elfarissi, and Z. Bougroun, "Machine Learning for Credit Card Fraud Detection," *Lect. Notes Electr. Eng.*, vol. 745, no. 24, pp. 211–221,

2022, doi: 10.1007/978-981-33-6893-4_20.

[11] E. F. Malik, K. W. Khaw, B. Belaton, W. P. Wong, and X. Chew, "Credit Card Fraud Detection Using a New Hybrid Machine Learning Architecture," *Mathematics*, vol. 10, no. 9, 2022, doi: 10.3390/math10091480.

[12] R. Bin Sulaiman, V. Schetinin, and P. Sant, "Review of Machine Learning Approach on Credit Card Fraud Detection," *Human-Centric Intell. Syst.*, vol. 2, no. 1–2, pp. 55–68, 2022, doi: 10.1007/s44230-022-00004-0.

[13] A. Mniai and K. Jebari, "Credit Card Fraud Detection by Improved SVDD," *Lect. Notes Eng. Comput. Sci.*, vol. 2244, pp. 32–37, 2022.

[14] B. Tayeb, A. Amine, H. M. Reda, and A. V. S. Kumar, "Credit Card Fraud Detection Using Deep Learning Approach (LSTM) Under IoT Environment," *Int. J. Organ. Collect. Intell.*, vol. 12, no. 1, pp. 1–20, 2022, doi: 10.4018/ijoci.305207.

[15] D. Prajapati, A. Tripathi, J. Mehta, K. Jhaveri, and V. Kelkar, "Credit Card Fraud Detection Using Machine Learning," *2021 7th IEEE Int. Conf. Adv. Comput. Commun. Control. ICAC3 2021*, pp. 421–427, 2021, doi: 10.1109/ICAC353642.2021.9697227.

[16] M. Habibpour *et al.*, "Uncertainty-Aware Credit Card Fraud Detection Using Deep Learning," no. July, 2021, [Online]. Available: http://arxiv.org/abs/2107.13508

[17] D. Bagchi, A. Mukherjee, and S. Pal, "A One Step Further Approach to Fraud Detection," vol. 2, no. 2, pp. 112–119, 2021.

[18] I. No, "Implementation of Credit Card Fraud," vol. 12, no. 06, pp. 163–175, 2021.

[19] H. Kochhar and Y. Chhabra, "A Novel Framework for Credit Card Fraud Detection," *Turkish J. Comput. Math. Educ.*, vol. 12, no. 11, pp. 3189–3195, 2021.

[20] Y. Xie, A. Li, L. Gao, and Z. Liu, "A Heterogeneous Ensemble Learning Model Based on Data Distribution for Credit Card Fraud Detection," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/2531210.

[21] S. Sanober *et al.*, "An Enhanced Secure Deep Learning Algorithm for Fraud Detection in Wireless Communication," *Wirel. Commun. Mob. Comput.*, vol. 2021, 2021, doi: 10.1155/2021/6079582.

[22] N. M. Mqadi, N. Naicker, and T. Adeliyi, "Solving Misclassification of the Credit Card Imbalance Problem Using near Miss," *Math. Probl. Eng.*, vol. 2021, 2021, doi:

10.1155/2021/7194728.

[23] A. Mehbodniya *et al.*, "Financial Fraud Detection in Healthcare Using Machine Learning and Deep Learning Techniques," *Secur. Commun. Networks*, vol. 2021, 2021, doi: 10.1155/2021/9293877.

[24] R. Chourasiya, "Employment Opportunities in Solar Energy Sector," *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 6, no. 1, pp. 1046–1053, 2021, doi: 10.48175/568.

[25] A. Bansal and H. Garg, "An Efficient Techniques for Fraudulent detection in Credit Card Dataset: A Comprehensive study," *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 1116, no. 1, p. 012181, 2021, doi: 10.1088/1757-899x/1116/1/012181.

[26] N. Uchhana, R. Ranjan, S. Sharma, D. Agrawal, and A. Punde, "Literature Review of Different Machine Learning Algorithms for Credit Card Fraud Detection," *Int. J. Innov. Technol. Explor. Eng.*, vol. 10, no. 6, pp. 101–108, 2021, doi: 10.35940/ijitee.c8400.0410621.

[27] A. S. Rathore, A. Kumar, D. Tomar, V. Goyal, K. Sarda, and D. Vij, "Credit Card Fraud Detection using Machine Learning," *Proc. 2021 10th Int. Conf. Syst. Model. Adv. Res. Trends, SMART 2021*, pp. 167–171, 2021, doi: 10.1109/SMART52563.2021.9676262.

[28] A. Mohari, J. Dowerah, K. Das, F. Koucher, and D. J. Bora, "Credit Card Fraud Detection Techniques: A Review," no. July, pp. 157–166, 2021, doi: 10.1007/978-981-16-1048-6_12.

[29] V. Muthulakshmi, C. Saravanakumar, and A. Tamizhselvi, "An Efficient Machine Learning Model for Location Aware Credit Fraud and Risk Classification and Detection," 2021, doi: 10.4108/eai.16-5-2020.2304200.

[30] . A., "Credit Card Fraud Detection using Machine Learning and Data Science," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. VII, pp. 3788–3792, 2021, doi: 10.22214/ijraset.2021.37200.

[31] I. Journal, E. Vol, I. Factor, and J. Homepage, "IJMIE13Jan21-NagAka," vol. 11, no. 01, pp. 108–112, 2021.

[32] G. Cabanac, C. Labbé, and A. Magazinov, "Tortured phrases: A dubious writing style emerging in science. Evidence of critical issues affecting established journals," 2021, [Online]. Available: http://arxiv.org/abs/2107.06751

[33] J. Liu, X. Gu, and C. Shang, "Quantitative Detection of Financial Fraud Based on Deep Learning with Combination of E-Commerce Big Data," *Complexity*, vol. 2020, 2020, doi: 10.1155/2020/6685888.

[34] Y. Yang, R. Chen, X. Bai, and D. Chen, "Finance fraud detection with neural network," *E3S Web Conf.*, vol. 214, pp. 4–7, 2020, doi: 10.1051/e3sconf/202021403005.

[35] A. M. Babu and A. Pratap, "Credit Card Fraud Detection Using Deep Learning," *2020 IEEE Recent Adv. Intell. Comput. Syst. RAICS 2020*, pp. 32–36, 2020, doi: 10.1109/RAICS51191.2020.9332497.

[36] Z. Zhang and S. Huang, "Credit Card Fraud Detection via Deep Learning Method Using Data Balance Tools," *Proc. - 2020 Int. Conf. Comput. Sci. Manag. Technol. ICCSMT 2020*, pp. 133–137, 2020, doi: 10.1109/ICCSMT51754.2020.00033.

[37] H. Najadat, O. Altiti, A. A. Aqouleh, and M. Younes, "Credit Card Fraud Detection Based on Machine and Deep Learning," *2020 11th Int. Conf. Inf. Commun. Syst. ICICS 2020*, no. Section IX, pp. 204–208, 2020, doi: 10.1109/ICICS49469.2020.239524.

[38] I. Karkaba, E. M. Adnani, and M. Erritali, "Deep Learning Detecting Fraud in Credit Card Transactions," *J. Theor. Appl. Inf. Technol.*, vol. 101, no. 9, pp. 3557–3565, 2023.

[39] R. Bin Sulaiman, V. Schetinin, and P. Sant, "Review of Machine Learning Approach on Credit Card Fraud Detection," *Human-Centric Intell. Syst.*, vol. 2, no. 1–2, pp. 55–68, 2022, doi: 10.1007/s44230-022-00004-0.

[40] V. Plakandaras, P. Gogas, T. Papadimitriou, and I. Tsamardinos, "Credit Card Fraud Detection with Automated Machine Learning Systems," *Appl. Artif. Intell.*, vol. 36, no. 1, 2022, doi: 10.1080/08839514.2022.2086354.

[41] A. Saxena, "Credit Card Fraud Detection using Machine Learning and Data Science," *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 10, no. 12, pp. 1890–1898, 2022, doi: 10.22214/ijraset.2022.48293.