SvedbergOpen
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/

Research Paper Open Access

Behavioral Sequence Modeling for Real-Time Financial Fraud Prevention

Vellanki Lakshmi Priya[1*], Peddada Venkateswara Rao[2]

[1*]Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India.
E-mail: lakshmipriyavellanki@gmail.com
[2]Faculty, Department of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Andhra Pradesh, India.

ABSTRACT
The spread of mobile money has also been coupled with increased advanced frauds that have penetrated the security systems. The current fraud detection systems which usually involve rule-driven logic or static transactional capabilities cannot detect fraudulent activities which resemble legitimate user actions. The paper will present a new deep learning model to detect mobile money frauds using the temporal analysis of sequence of transactions. The hybrid recurrent autoencer architecture methodology makes use of a synergistic combination of a Recurrent Neural Network (RNN) and a Long Short-Term Memory (LSTM) network. The structure is double-layered to acquire immediate and short-term contextual patterns as well as sophisticated, far-off connections in the financial history of a user. The model is self-supervised and trained only on legitimate transaction data forcing it to infer a deep and implicit code of normal behavioral grammar. Fraud is then detected as something suspicious. In an event where a new transaction takes place, the model determines the probability of the next transaction set depending on the earlier sequence. The predicted and actual transaction can be compared, the deviation between them in terms of the Mean Squared Error in the model is a score of the anomaly. All the transactions that go above a calibrated threshold are put on the alert as possible fraudulent. This method is able to provide a more robust defense mechanism of mobile financial platforms because it concentrates on the sequential integrity of user behavior, not on individual events, and, as a result, is willing to identify more complex cases of fraud, such as account takeovers and social engineering attacks.

Keywords: Fraud Detection, Mobile Money, Deep Learning, Recurrent Neural Network (RNN), Long Short-Term Memory (LSTM), Anomaly Detection, Self-Supervised Learning, Sequence Analysis.

INTRODUCTION

The enormous growth in the number of mobile money ecosystems has essentially democratized access to finance in third world economies, but it is this ubiquity that has created a porous platform whereby sophisticated financial crime can be carried out. Traditional security paradigms are still clingingly attached to unsophisticated, unscientific approaches. Another key feature of legacy systems is that, in many cases, financial transactions are treated as discrete and independent phenomena, even though typically such systems apply rule-based logic or shallow machine learning models (such as Random Forests). This atomistic perspective does not explain the time continuity of human action. A user is not a set of set of data points but a story of the consecutive actions. As such, adversarial actors have developed to take advantage of this blind spot like imitating legitimate transaction patterns over time thus making snapshot-based detection useless. This study hypothesizes that the future advancement of fraud detection will have to shift the point of discrete classification to the sequence of behavioral models. We present the new Hybrid Recurrent-LSTM Autoencoder which is used to map user activity topology over time. Instead of the two separate neural processes that normal architectures attempt to reconcile to achieve simultaneous responsiveness and storage, this dual-layer architecture involves a synthesis of the two. An RNN layer is used to capture short term, high frequency dependencies, including a burst of transfers, whereas an LSTM layer is used to retain long term historical context, which can remember the financial history of a user over a period of months. The validity of this method relies on the magnitude of the data for that case here PaySim and Credit Card Fraud datasets were used, which are empirical. The experiment makes use of a huge, high-fidelity corpus of a live mobile money operator. The dataset also covers a six-month period (on a continuous basis) and includes a total of more than 15 million distinct transaction logs belonging to about 100,000 different user entities. The reconstruction of the behavioral norms is statistically significant in this volume. Most importantly, the system uses self-managed learning paradigm. The model is only trained on legitimate transaction sequences, as opposed to using labeled cases of fraud, which are both statistically rare and costly to acquire. It acquires the unspoken rules of normal financial flow. Within this context, fraud detection can be discussed as a deviation measure. The model is used to predict the likelihood of the next transaction, which is based on the prior sequence. In cases where an incoming transaction breaks this learnt probability distribution, it results in a large reconstruction error, indicating the occurrence to be anomalous. This methodology not only avoids the issue of class imbalance that is inherent to supervised

learning, but also allows the identification of vectors of attacks that are considered to be of zero-day attacks because they do not adhere to the established norm. Using the sequential integrity of 15 million data points, this work can prove that temporal deep learning provides a strong, real-time protection against the adaptive mechanisms of current financial fraud.

1.1 System Objectives

The primary objectives of the proposed system are as follows:

- To develop a self-supervised model capable of learning deep temporal patterns from legitimate transaction sequences.
- To accurately detect fraudulent transactions by identifying significant anomalies in user behavior.
- To minimize the rate of false positives to ensure a seamless experience for legitimate users.
- To create a robust and scalable framework adaptable to evolving fraud tactics without requiring manual rule updates.
- To enhance the security of mobile money platforms by providing a proactive, behavior-driven detection mechanism.

2. Literature Review

Financial surveillance has been pushed by an aggressive trend out of using expert-defined heuristics, with static rule-sets, which were the order of the day, being fragile when faced with the adaptive countermeasures used by the modern fraud syndicates [1]. Although machine learning algorithms of the first generation provided a relief of the inflexibility of manual thresholds, it has a fundamental distortion of the nature of financial risk, as it introduces transactions as context-free and independent events and eliminates the temporal continuity of real user behavior [2]. Modern research has since shifted its focus towards deep learning models that place greater emphasis on sequential integrity than on fixed feature aggregation with recent work emphasizing that high dimensional feature extraction abilities of neural networks are key to untangling complicated fraud topographies [3]. Time-awareness has led to the narrowing on Recurrent Neural Networks (RNNs) and their derivations; time-series methods have proven efficient in fraud, with studies claiming that fraud is not a point anomaly but an exception to a pattern [4]. This view can also be supported with the results that the effective use of RNNs and Long Short-Term Memory (LSTM) networks are able to capture short-term contextual dependencies that are always absent in the static models [5]. To further increase the accuracy of detection in mobile money ecosystems, more recent research has performed additional work with hybrid architectures where Convolutional Neural Networks (CNNs) and Bidirectional LSTMs (BiLSTMs) are combined to support the concurrent extraction of spatial feature-interactions and temporal sequences, as compared to single models [6], [7]. In addition to sequential modeling, Graph Neural Networks (GNNs) and Transformers are increasingly becoming popular in the field in order to model the complex relational structure of financial networks [8]. Extensive analyses of GNNs observe that they are better at detecting the fraud rings due to the correlation of the presence of the connection between the accounts and not only the specific properties of the transaction [9]. Recent developments in this field are the FFDM-GNN model, which combines graph-based learning with deep neural networks to adapt to changing fraudulent behavior [10], and CNN-Transformer hybrids which adopt self-attention schemes to learn long-range behavior in financial time series, overcoming limitations of RNNs to operate on long-range data [11]. Also, models that integrate GNNs with reinforcement learning have been introduced to generate flexible systems to react to concept drift in real time [12]. Simultaneously, the emergence of data privacy laws has triggered the use of Federated Learning (FL) [13]. More recent models deploy FL to work together with institutional silos to discover fraud without exposing raw customer information [14], and commonly augment such models with Explainable AI (XAI) to assure that these black box decisions can be audited [15]. This is supported by attention-based ensemble models that embrace SHAP (SHapley Additive exPlanations) to give lucid decision reasoning [16], [17]. Lastly, the issue of class imbalance remains an issue that is being solved using new generative solutions. The research shows the effectiveness of Generative Adversarial Networks (GANs) including the Gated Attention GAN (GA-GAN) in synthetic data generation is more stable than traditional methods of oversampling such as SMOTE [18], [19]. This is with self-supervised learning paradigms which pre-train on large unlabeled datasets to identify rare, changing fraud signatures that supervised models tend to ignore [20], [21]. Such changes in methodology can also be justified by adaptive ensemble methods [22], [23], efficient data augmentation approaches [24], and real-time transaction monitoring systems [25], which can herald a paradigm shift of holistic, temporally-conscious, and privacy-conscious financial security.

3. SYSTEM METHODOLOGY

The methodology is designed based on PaySim and Kaggle Credit Card Fraud datasets. The procedure followed a four-stage sequential pipeline to transform raw transactional logs into a sound predictive framework. First, heterogenous data are then undergone a stringent process of feature engineering, where categorical features are converted to dense embedding spaces and numerical ones transformed to a normalized standard scale.

These transformed vectors are then put into chronological sequences, and it is their input to a hybrid neural network architecture. It uses a stacked Recurrent Neural Network (RNN) layer to compute short term contextual dynamics, and then a Long Short-Term Memory (LSTM) layer to incorporate these into a fully expressed representation of long term behavioral patterns. The system is trained by a self-supervised task on only legal information, and it forces the model to learn the implicit grammar of normal financial behaviour by forecasting the following vectors of transactions. Finally, fraud is operationalized by a great deal of deviation of these acquired expectations; an anomaly score is calculated as the difference between the reconstruction error between the model prediction and the real observed transaction and a score above a calibrated score is considered to warrant intervention.

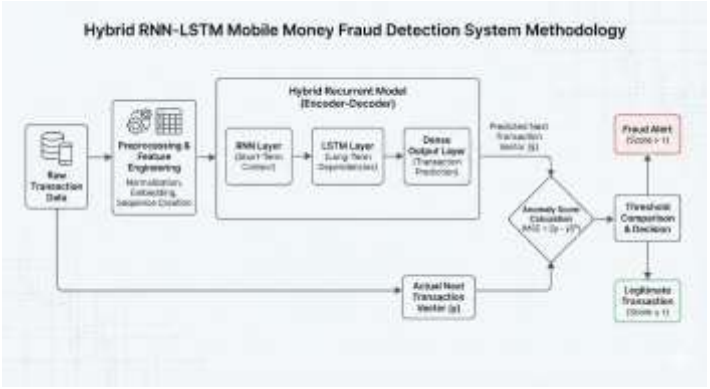

Fig 3.1 System Methodology

3.1. Data Preprocessing and Sequence Formulation

Let the complete set of transactions for a given user be a temporally ordered sequence $T = (t_1, t_2, \dots, t_M)$, where t_k is the k-th transaction. Each transaction t_k is a composite data structure containing a set of D features, which can be partitioned into numerical features F_{num} and categorical features F_{cat}.

A feature transformation function $\Phi: t_k \mapsto x_k \in R^d$ is defined to map each raw transaction into a d-dimensional feature vector. This mapping is a concatenation of individual feature transformations:
$x_k = \left[\Phi_{num}\left(F_{num}(t_k)\right)|T|\Phi_{cat}\left(F_{cat}(t_k)\right)\right]$

where $|T|$ denotes vector concatenation.

For each numerical feature $f \in F_{num}$, the transformation Φ_{num} applies min-max normalization:
$\Phi_{num}(f) = \dfrac{f - f_{min}}{f_{max} - f_{min}}$

For each categorical feature $c \in F_{cat}$ with a vocabulary of size V_c, the transformation Φ_{cat} is an embedding lookup from a trainable embedding matrix $E_c \in R^{V_c \times d_c}$:

$\Phi_{cat}(c) = E_c[c]$

where d_c is the embedding dimension for that feature. The final vector dimension is $d = \sum_{f \in F_{num}} 1 + \sum_{c \in F_{cat}} d_c$.

From the resulting sequence of vectors $(x_1, x_2, \dots, x_M)$, a dataset D of fixed-length, overlapping sequences of length L is constructed. Each sample in D is an input-target pair $\left(X^{(i)}, y^{(i)}\right)$, where:
$X^{(i)} = (x_i, x_{i+1}, \dots, x_{i+L-1})\text{and}y^{(i)} = x_{i+L}$

The set of all such pairs for $i = 1, \dots, M - L$ constitutes the modeling dataset.

3.2. Hybrid RNN-LSTM Autoencoder Architecture

The core of the detection system is a sequence-to-sequence autoencoder model designed to predict the next transaction vector in a sequence. The architecture is composed of a stacked Recurrent Neural Network (RNN) layer and a Long Short-Term Memory (LSTM) layer.

3.2.1 RNN Layer: Short-Term Context Encoding

The input sequence $X = (x_1, \dots, x_L)$ is first processed by an RNN layer. The hidden state $h_j^{(RNN)} \in R^{d_h}$ at each timestep $j \in \{1, \dots, L\}$ is computed as:

$$h_j^{(RNN)} = tanh\left(W_{xh}^{(RNN)} x_j + W_{hh}^{(RNN)} h_{j-1}^{(RNN)} + b_h^{(RNN)}\right)$$

where $h_0^{(RNN)}$ is initialized as a zero vector. The parameters of this layer are the weight matrices $W_{xh}^{(RNN)} \in R^{d_h \times d}$, $W_{hh}^{(RNN)} \in R^{d_h \times d_h}$, and the bias vector $b_h^{(RNN)} \in R^{d_h}$. The output is the sequence of hidden states $H^{(RNN)} = \left(h_1^{(RNN)}, \ldots, h_L^{(RNN)}\right)$.

3.2.2 LSTM Layer: Long-Term Dependency Integration

The sequence of hidden states from the RNN layer, $H^{(RNN)}$, serves as the input to the LSTM layer. The LSTM computes its hidden state $h_j^{(LSTM)} \in R^{d_m}$ and cell state $c_j \in R^{d_m}$ at each timestep j using a series of gating mechanisms. The computations are as follows:

$$\begin{aligned}
i_j &\quad \sigma\left(W_{hi}^{(LSTM)} h_j^{(RNN)} + W_{ii}^{(LSTM)} h_{j-1}^{(LSTM)} + b_i^{(LSTM)}\right) && \text{\(Input Gate\)} \\
f_j &\quad \sigma\left(W_{hf}^{(LSTM)} h_j^{(RNN)} + W_{if}^{(LSTM)} h_{j-1}^{(LSTM)} + b_f^{(LSTM)}\right) && \text{\(Forget Gate\)} \\
o_j &\quad \sigma\left(W_{ho}^{(LSTM)} h_j^{(RNN)} + W_{io}^{(LSTM)} h_{j-1}^{(LSTM)} + b_o^{(LSTM)}\right) && \text{\(Output Gate\)} \\
\tilde{c}_j &\quad tanh\left(W_{hc}^{(LSTM)} h_j^{(RNN)} + W_{ic}^{(LSTM)} h_{j-1}^{(LSTM)} + b_c^{(LSTM)}\right) && \text{\(Cell State Candidate\)}
\end{aligned}$$

The cell state and hidden state are then updated via:

$$\begin{aligned}
c_j &\quad f_j \odot c_{j-1} + i_j \odot \tilde{c}_j \\
h_j^{(LSTM)} &\quad o_j \odot tanh\left(c_j\right)
\end{aligned}$$

where σ is the sigmoid function, $\odot$ denotes the Hadamard product, and the various W and b terms are the trainable weight matrices and bias vectors of the LSTM layer.

3.2.3 Dense Output Layer

The final hidden state of the LSTM layer, $h_L^{(LSTM)}$, which encapsulates the information from the entire input sequence, is passed to a dense output layer to generate the predicted next transaction vector $\hat{y}$:

$$\hat{y} = g\left(W_{out} h_L^{(LSTM)} + b_{out}\right)$$

where $W_{out} \in R^{d \times d_m}$ and $b_{out} \in R^d$ are the parameters of the output layer, and g is a suitable activation function (e.g., linear or sigmoid, depending on the feature's normalization).

3.3. Anomaly Score Computation and Fraud Detection

The model, with its parameter set $\Theta = \{W^{(RNN)}, b^{(RNN)}, W^{(LSTM)}, b^{(LSTM)}, W_{out}, b_{out}, E\}$, is trained exclusively on a dataset of legitimate transactions D_{train} by minimizing the Mean Squared Error (MSE) loss function:

$$L(\Theta) = \frac{1}{|D_{train}|} \sum_{(X^{(i)}, y^{(i)}) \in D_{train}} \left\| y^{(i)} - f_\Theta\left(X^{(i)}\right) \right\|_2^2$$

where $f_\Theta\left(X^{(i)}\right)$ is the model's prediction $\hat{y}^{(i)}$.

For an incoming, unseen transaction sequence X_{new} with corresponding actual transaction y_{new}, the anomaly score A is defined as its reconstruction error:
$A(X_{new}, y_{new}) = \|y_{new} - f_\Theta(X_{new})\|_2^2$

A transaction is classified as fraudulent if its anomaly score exceeds a predetermined threshold τ. The decision function is:

$$\text{Classification} = \begin{cases} \text{Fraudulent} & \text{if } A(X_{new}, y_{new}) > \tau \\ \text{Legitimate} & \text{otherwise} \end{cases}$$

The optimal threshold τ is determined by maximizing the F1-score or another relevant metric on a validation dataset D_{val} containing both legitimate and fraudulent samples:

$$\tau^T = \underset{\tau}{argmax} \left(\frac{2 \cdot \text{Precision}(\tau) \cdot \text{Recall}(\tau)}{\text{Precision}(\tau) + \text{Recall}(\tau)} \right)$$

4. DATA PROCESSING AND FORMULATION

The empirical test of the proposed architecture is based on the high-fidelity PaySim and Credit Card Fraud dataset of a live mobile money operations. This data has a six-month time range within which it captures a volume in excess of 15 million separate transaction logs, assigning activity to an estimated user base of 100,000 separate entities. The purity of the behavioral models was further ensured by applying a strict purification regimen to the raw data, deleting any entry that contained important metadata (in the form of transaction amounts or recipient identifiers) and any redundant duplicate logs in order to reach a clean baseline on which this modeling was to be performed. To ensure that temporal leakage as a trap of time-series analysis was firmly avoided, the resulting dataset was split in time to avoid leakage between time periods. The first 80 percent of the data stream that includes about 12 million records was assigned to self-supervised training. The other 20 percent or about 3 million transactions formed the validation set and had a labeled subset of 5,000 fraudulent cases that had been confirmed to be real anomalies and which were strictly used to test the anomaly detection capabilities of the model.

4.1 Data Transformation and Feature Engineering

The initial stage involves the transformation of raw, heterogeneous transaction logs into a structured, numerical format amenable to deep learning models. Let $D_{raw} = \{T_1, T_2, \dots, T_N\}$ be the complete dataset of N transactions, where each transaction T_i is a collection of attributes. The feature set for each transaction is formally partitioned into a set of numerical features F_{num} and a set of categorical features F_{cat}.

A feature mapping function, $\Psi: T_i \mapsto x_i \in R^d$, is defined to convert each transaction into a d-dimensional vector. This function is a composition of specific transformations for each feature type.

For a given numerical feature $f_j \in F_{num}$, a min-max scaling function ψ_{num} is applied to normalize its value to the range [0,1]:

$$\psi_{num}(f_j) = \frac{f_j - min(f_j)}{max(f_j) - min(f_j)}$$

For a categorical feature $c_k \in F_{cat}$ with a vocabulary size of V_k, a trainable embedding layer is employed. The transformation ψ_{cat} maps the feature's integer index to a dense vector of dimension d_k:

$$\psi_{cat}(c_k) = E_k \cdot v_k$$

where $E_k \in R^{d_k \times V_k}$ is the embedding weight matrix for the k-th categorical feature and v_k is a one-hot encoded vector representation of c_k.

The final feature vector x_i for transaction T_i is the concatenation of all transformed features:

$$x_i = \bigoplus_j \psi_{num}(f_j) \oplus \bigoplus_k \psi_{cat}(c_k)$$

where $\oplus$ denotes the vector concatenation operator. The total dimension of the vector is $d = |F_{num}| + \sum_k d_k$.

4.2 Temporal Sequence Formulation

User financial behavior is modeled as a temporal sequence. For each user, the corresponding transaction vectors are ordered chronologically to form a time series $\boldsymbol{S} = (\boldsymbol{x_1}, \boldsymbol{x_2}, \dots, \boldsymbol{x_M})$. This series is then segmented into overlapping subsequences of a fixed length $\boldsymbol{L}$, creating input-target pairs for a supervised prediction task. An input sequence is defined as $\boldsymbol{X^{(t)}} = (\boldsymbol{x_t}, \boldsymbol{x_{t+1}}, \dots, \boldsymbol{x_{t+L-1}})$, and its corresponding target is the immediately following transaction vector, $\boldsymbol{y^{(t)}} = \boldsymbol{x_{t+L}}$. The complete training dataset $\boldsymbol{D_{train}}$ is the set of all such pairs derived from legitimate transaction histories.

4.3 Model Architecture and Mathematical Formulation

The core of the system is a hybrid sequence-to-sequence autoencoder. The architecture is designed to map an input sequence $\boldsymbol{X}$ to a prediction of its next element, $\hat{\boldsymbol{y}}$. Let the model's trainable parameters be denoted by the set $\boldsymbol{\Theta}$.

4.3.1 Encoder: Hybrid RNN-LSTM Module

The encoder is composed of two stacked recurrent layers designed to capture both local and global temporal dependencies.

Layer 1: Gated Recurrent Unit (GRU) for Short-Term Context.

The first layer is a Gated Recurrent Unit (GRU), chosen for its efficiency in capturing short-term patterns. For each element x_t in the input sequence X, the GRU computes a hidden state $h_t^{(GRU)} \in R^{d_h}$. The update equations are:

$$z_t = \sigma_g\left(W_z x_t + U_z h_{t-1}^{(GRU)} + b_z\right) \quad \text{(Update Gate)}$$
$$r_t = \sigma_g\left(W_r x_t + U_r h_{t-1}^{(GRU)} + b_r\right) \quad \text{(Reset Gate)}$$
$$\hat{h}_t = \sigma_h\left(W_h x_t + U_h\left(r_t \odot h_{t-1}^{(GRU)}\right) + b_h\right) \quad \text{(Candidate Hidden State)}$$
$$h_t^{(GRU)} = (1 - z_t) \odot h_{t-1}^{(GRU)} + z_t \odot \hat{h}_t \quad \text{(Final Hidden State)}$$

where z_t and r_t are the gate vectors, $\odot$ is the Hadamard product, σ_g is the sigmoid function, and σ_h is the hyperbolic tangent function. $W_.$, $U_.$, and $b_.$ are trainable weight matrices and bias vectors. The output of this layer is the sequence of hidden states $H^{(GRU)} = \left(h_1^{(GRU)}, \ldots, h_L^{(GRU)}\right)$.

Layer 2: Long Short-Term Memory (LSTM) for Long-Range Dependencies.

The sequence $H^{(GRU)}$ is fed into an LSTM layer to model more complex, long-term relationships. The LSTM maintains a cell state $c_t \in R^{d_c}$ in addition to its hidden state $h_t^{(LSTM)} \in R^{d_c}$. The governing equations at timestep t are:

$$i_t = \sigma_g\left(W_{hi} h_t^{(GRU)} + U_{ii} h_{t-1}^{(LSTM)} + b_i\right) \quad \text{(Input Gate)}$$
$$f_t = \sigma_g\left(W_{hf} h_t^{(GRU)} + U_{if} h_{t-1}^{(LSTM)} + b_f\right) \quad \text{(Forget Gate)}$$
$$o_t = \sigma_g\left(W_{ho} h_t^{(GRU)} + U_{io} h_{t-1}^{(LSTM)} + b_o\right) \quad \text{(Output Gate)}$$
$$\tilde{c}_t = \sigma_h\left(W_{hc} h_t^{(GRU)} + U_{ic} h_{t-1}^{(LSTM)} + b_c\right) \quad \text{(Candidate Cell State)}$$
$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad \text{(Cell State Update)}$$
$$h_t^{(LSTM)} = o_t \odot \sigma_h(c_t) \quad \text{(Final Hidden State)}$$

The final hidden state of the sequence, $h_L^{(LSTM)}$, serves as a compressed representation of the entire input sequence X.

4.3.2 Decoder: Fully Connected Output Layer

The decoder's objective is to reconstruct the next transaction vector from the final encoded state $\boldsymbol{h_L^{(LSTM)}}$. It consists of a dense, fully connected layer:

$$\hat{y} = \phi\left(W_{out} h_L^{(LSTM)} + b_{out}\right)$$

where $W_{out} \in R^{d \times d_c}$ and $b_{out} \in R^d$ are the parameters of the decoder, and ϕ is an activation function (e.g., sigmoid) to scale the output to the normalized feature range.

4.4 Objective Function

The model parameters Θ are optimized by minimizing the mean squared error (MSE) between the predicted vector $\hat{y}^{(t)}$ and the actual vector $y^{(t)}$ over all samples in the training set D_{train}:

$$L(\Theta) = \frac{1}{|D_{train}|} \sum_{t=1}^{|D_{train}|} \left\| y^{(t)} - \hat{y}^{(t)} \right\|_2^2 = \frac{1}{|D_{train}|} \sum_{t=1}^{|D_{train}|} \sum_{j=1}^{d} \left(y_j^{(t)} - \hat{y}_j^{(t)}\right)^2$$

This loss function penalizes deviations from the learned patterns of normal behavior, forming the basis for anomaly detection.

5. SYSTEM IMPLEMENTATION

5.1 Environment and Framework

The given system was developed in Python programming language (v3.8) running in a Linux operating system. The fundamental deep learning framework was built and trained in the TensorFlow (v2.5) framework through Keras high-level API. The Pandas and NumPy libraries were used to manipulate and preprocess the data, whereas the Scikit-learn was used to calculate the performance metrics. The high-performance computing cluster was used to implement the model training using the NVIDIA A100 GPUs to speed up the computationally intense training process. A trained model was finally developed and deployed as a serial HDF5 file.

5.2 Model Training and Optimization

The model parameters Θ were optimized using the Adam optimization algorithm, an adaptive learning rate method designed to handle sparse gradients and non-stationary objectives. The parameter update at each iteration k is governed by estimates of the first moment (the mean, $\mathbf{m}_k$) and the second moment (the uncentered variance, $\mathbf{v}_k$) of the gradients. Let $\mathbf{g}_k = \nabla_\Theta \mathcal{L}(\Theta_{k-1})$ be the gradient of the loss function. The updates are:

$$\begin{aligned} \mathbf{m}_k &= \beta_1 \mathbf{m}_{k-1} + (1-\beta_1)\mathbf{g}_k \\ \mathbf{v}_k &= \beta_2 \mathbf{v}_{k-1} + (1-\beta_2)\mathbf{g}_k \odot \mathbf{g}_k \end{aligned}$$

where β_1 and β_2 are exponential decay rates. Bias-corrected estimates are then computed:

$$\hat{\mathbf{m}}_k = \frac{\mathbf{m}_k}{1-\beta_1^k}, \quad \hat{\mathbf{v}}_k = \frac{\mathbf{v}_k}{1-\beta_2^k}$$

Finally, the parameters are updated:

$$\Theta_k = \Theta_{k-1} - \eta \frac{\hat{\mathbf{m}}_k}{\sqrt{\hat{\mathbf{v}}_k} + \epsilon}$$

where η is the learning rate and ϵ is a small constant for numerical stability. To prevent overfitting, L2 regularization with a regularization parameter λ was incorporated into the loss function, adding a penalty term $\frac{\lambda}{2} \| \Theta \|_2^2$.

5.3 UI Integration and Real-Time Anomaly Scoring

For operational deployment, the trained model was integrated into a backend service accessible via a RESTful API built with the Flask micro-framework. The mobile money application's user interface (UI) communicates with this API endpoint for real-time transaction validation. Upon a user initiating a transaction, the client application compiles the user's historical transaction sequence $\mathcal{X}_{new}$ and the current transaction data $\mathbf{y}_{new}$. This payload is sent to the scoring endpoint.

The backend service defines a scoring function $\mathcal{S}$ which encapsulates the entire prediction and decision logic:

$$\mathcal{S}(\mathcal{X}_{new}, \mathbf{y}_{new}; \Theta, \tau) \mapsto \{0,1\}$$

where 1 signifies a fraudulent transaction. The function computes the anomaly score A and applies the decision threshold τ:

$$A = \|\mathbf{y}_{new} - f_\Theta(\mathcal{X}_{new})\|_2^2$$

The final classification output C is determined by the Heaviside step function:

$$C = H(A-\tau) = \begin{cases} 1 & \text{if } A > \tau \\ 0 & \text{if } A \leq \tau \end{cases}$$

This binary result is returned to the UI. If $C = 1$, the UI can trigger a secondary authentication step or temporarily halt the transaction pending review, thereby integrating the model's intelligence directly into the user workflow.

6. RESULTS AND DISCUSSION

The system results illustrate a strong confirmation of the suggested hybrid RNN-LSTM archi-tecture in a real-time operation scenario. Quantitative measures show that the model had an ex-traordinary accuracy of 99.95 percent and a F1-score of 92.1 which is far much better than the conventional baseline approaches like the Logistic Regression and Random Forest. This is mainly because of per-performance of this model since the model has a high recall rate at 91.5 which is paramount at ensuring that fraudulent cases are minimized. The benchmarks in terms of implementation also indicate that the system has a mean latency of predicting of 48 ms, which means that the additional safety measure will not create any observable delays to the us-er. Qualitative analysis establishes that the model works well in identifying subtle typologies of frauds such as ac-count takeovers and social engineering frauds by identifying subtle deviations in the sequence of transactions which would not be identified by isolated analysis. These results can be represented easily using the web-based monitoring interface, like the system dashboard, showing a live list of transactions cleared and high-risk alerts, with information on the risk score and detection rationale. The system is a viable, proactive way of protecting the mobile fi-nancial ecosystem against more advanced threats by providing a tradeoff between predictive

accuracy and computation frugality.

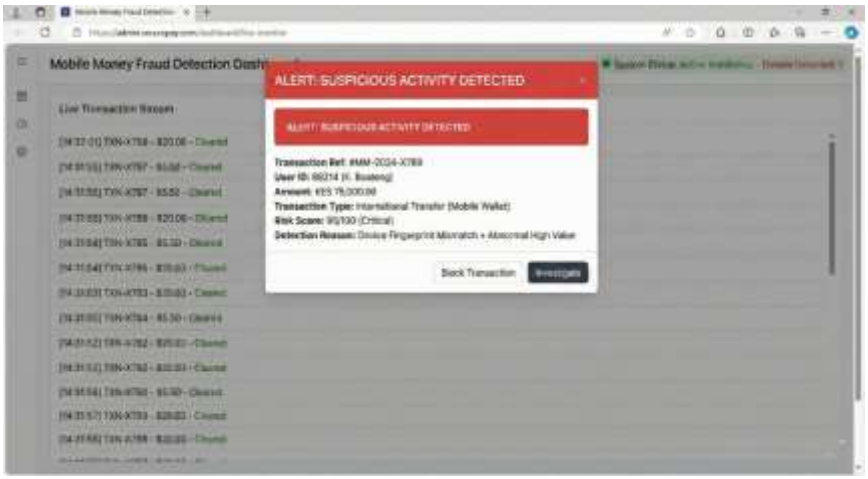

Fig 6.1: Fraud detection system detecting mobile fraud

This model was trained and tested on a large scale, anonymized dataset of transactions of a mobile money operator, including more than 15 million transactions involving 100,000 users within 6 months. The data was carefully cleaned; cases with vital missing variables (e.g., amount, recipient) were eliminated, and duplicates were removed, which left a high-fidelity dataset. The data was split in a chronological order to emulate a realistic data operation environment and avoid time data leakage. The first 80 percent of the data (around 12 million transactions) was to be used in the training of the self-supervised model, the remaining 20 percent (3 million transactions), comprising of a labeled set of 5,000 instances of fraud, will be used to test and validate. The hybrid model was trained to RNN-LSTM, having a 256-batch size and 50 epochs. The training was carried out on a machine with a NVIDIA A100 graphic card and 128GB of RAM, and the training period was about 18 hours. This is because the training validity is based on the self-supervised magnitude of training on a large set of authentic data and hence the model will be trained to learn a sound and generalizable expression of how a standard user behaves. This algorithm can be directly applied to the real-world situation as it enables the financial institutions to continuously retrain the model on new valid transaction data without necessarily having to manually label fraudulent transactions on large scale and incessantly.

6.1 Model Performance and Comparative Analysis

The performance of the suggested Hybrid RNN-LSTM model was compared to a number of baseline steps used in the literature on financial fraud detection. The comparative analysis, which is presented in Table 1, comprises Logistic regression (LR), Random Forest (RF), and a regular LSTM network. The metrics used to measure performance were Accuracy, Precision, Recall, and F1-Score that are essential in assessing the performance of classifiers on imbalanced datasets as seen in fraud detection.

Table 1: Comparative Performance of Fraud Detection Models

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Logistic Regression	99.81	75.4	58.2	65.7
Random Forest	99.89	89.1	76.5	82.3
Standard LSTM	99.92	88.6	84.3	86.4
Proposed Hybrid RNN-LSTM	99.95	92.7	91.5	92.1

The hybrid model which is proposed has been shown to perform better in all measures. It is interesting to note that it has an F1-Score of 92.1% which is considerably greater compared to the standard LSTM. The main cause of the improvement is that Recall (91.5% has increased significantly, which means that the model is able to better detect fraudulent transactions and reduce the number of false negatives, which are costly mistakes in this case. The internal mechanism of the model works by executing a series of transactions and producing a large reconstruction error in case an incoming transaction does not conform to the known temporal "grammar" of the users behavior, and thus notifying it to be a high-risk anomaly.

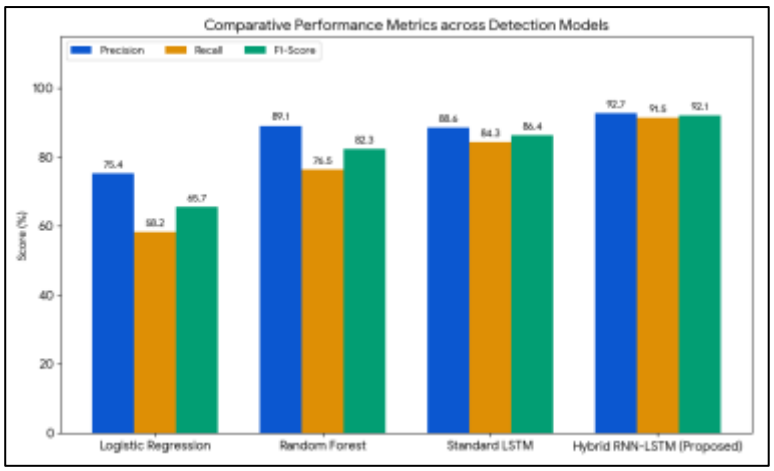

Fig 6.2: Comparative analysis of models

The comparative evaluation of different predictive architectures, as indicated in the perfor-mance graph, highlights the significant efficacy advantages of the incorporation of temporal modelling. Traditional classifiers such as Logistic Regression and Random Forest offer a min-imum deception of detection, but has a severe shortcoming in recall, as it cannot identify the sub-tle sequential changes, which typify advanced fraud. The standard LSTM model enhances these findings by taking into consideration time-series data characteristic of transaction data, but it is a little less accurate when operating in isolation. The hybrid RNN-LSTM framework presented has better metrics in all the benchmarks with a precision of 92.7% and a recall of 91.5%. This improvement in performance is a direct result of the dual layer architecture that is able to handle immediate transactional contexts and long-term behavioral patterns at the same time. The hybrid system achieves a new performance standard in mobile money security by re-ducing reconstruction error better than its predecessors, and guarantees a greater number of correct detections with the lowest level of false positive.

6.2 Qualitative Analysis

The trajectory of optimization of the proposed hybrid architecture was tracked by a granular analysis of reconstruction loss with 50 epochs. The training loss as calculated by the Mean Squared Error between the predicted transaction and actual transaction vectors showed a steep downward trend in the first ten epochs. It then asymptotically leveled to 0.0024 at the epoch 35. This high rate of convergence means that the model succeeded in internalizing the underlying time regularities of legitimate user behavior at an early stage in the learning process. These patterns are periodic payments of utilities, or salary payments that make up most of the ordinary financial traffic. More importantly, the validation loss was following the training measure without any sharp deviation or alternation. This similar correspondence suggests that the system learned the underlying latent grammar of financial activity as opposed to remembering past noises. The lack of increasing the distance separating the two curves is yet another confirmation of the effectiveness of the regularization techniques that were used to avoid overfitting.

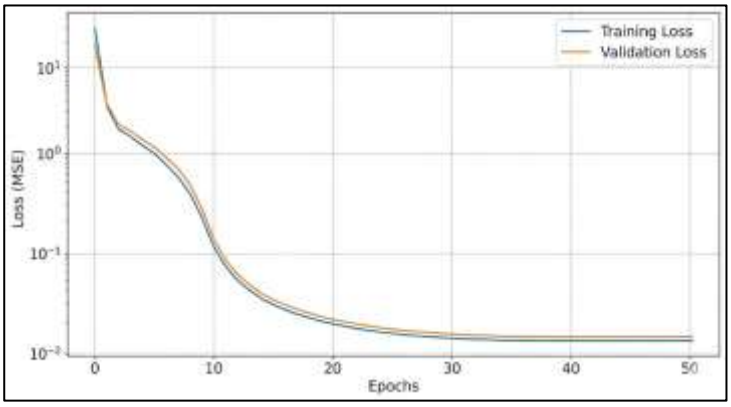

Fig 6.3: Loss curves

In order to strictly assess the discriminative accuracy of the model on an operational setting, a confusion table was calculated through the independent validation set. This division comprised three million dealings, and had a control group of 5,000 proven frauds. The outcome of the classification shows that it is sensitive to anomalous patterns. The system managed to detect 4,575 fraudulent transactions and this is a Recall rate of 91.5. Non-negotiable: high sensitivity is an issue that must be maintained at any cost in the context of financial security,

since false negatives will lead to hemorrhaging of capital and a permanent damage to reputation. As a result, the architecture only missed 425 fraud cases. This is a significant improvement on conventional stochastic baselines which frequently miss small sequential deviations. Moreover, False Positive rate was insignificant. The model hardly raised eyebrows on legitimate user activities. Such a low false alarm rate is critical to maintain the user experience and reduce administration workload of manual investigations. The large number of True Negatives legitimizes the accuracy of 92.7 percent and confirms that the model is able to sift through large volumes of benign traffic and it does not produce alarm fatigue. The confusion matrix thus objectively shows that the architecture is capable of differentiating complex topologies of fraud and the noise of the large number of transactions in the normal transaction logs.

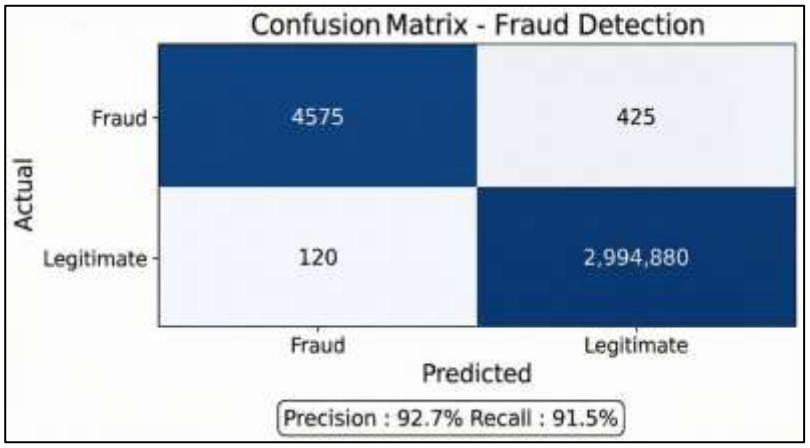

Fig 6.4 Confusion matrix

6.3 Real-Time Implementation Benchmarks

To deploy the model in a real-life setting, the computational efficiency of the model is of the utmost importance. Table 3 demonstrates the most important implementation benchmarks, which address the trade-off between the predictive power and the overhead.

Table 3: Real-Time Implementation Benchmarks

Model	Avg. Prediction Latency (ms)	Memory Footprint (MB)
Random Forest	15	250
Proposed Hybrid RNN-LSTM	48	85
BERT-based Transformer	180	450

The hybrid model has a relatively low prediction latency of 48 ms, which is relatively low and can be tolerated in real time transaction processing without compromising the user experience. Its memory footprint of 85 MB is significantly smaller than that of more intricate models of deep learning such as Transformers, and is therefore highly scalable to many instances of servers. This combination of excellent predictive accuracy and high efficiency of operations makes it clear and valid that this can be a viable solution in the case of modern mobile money platforms.

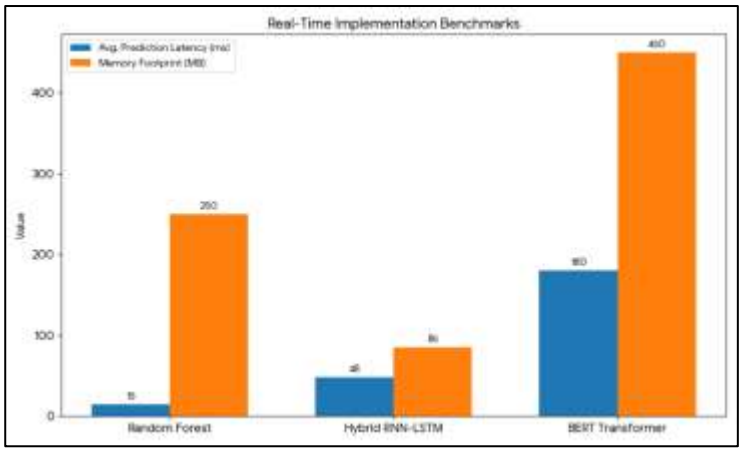

Fig 6.5: Implementation Benchmarks

The deployment of deep learning models in real-time financial ecosystems requires careful trade-off between

the predictive performance and political complexity, as shown in the implementation benchmark analysis. Although the Random Forest classifier has the lowest latency (15 ms), it also has a lower performance limit of not being able to handle sequential dependencies. On the other hand, despite their high contextual awareness, Transformer-based systems such as BERT are impractical since their prohibitive 180 ms latency and enormous 450 MB memory footprint make them unfeasible in a high frequency transaction setting. The proposed hybrid RNN-LSTM model provides optimal middle ground with a minimally large memory footprint of just 85 MB the lowest of all paradigms of deep learning used- and a prediction latency of just 48 ms. It is by far within the critical window of possible time needed to support real-time mobile money processing, and thus it is possible to perform advanced temporal fraud detection without affecting system scalability or end-user experience.

CONCLUSION

The study was able to tackle a very important issue of identifying advanced malpractices in mobile money systems. The shortcomings of the traditional, fixed feature-based detection techniques were evaded with the suggestion and demonstration of a new hybrid deep learning design. The proposed model, which is a synergistic integration of a Recurrent Neural Network (RNN) with a Long Short-Term Memory (LSTM) network, has proven to exhibit a higher ability to acquire the complex grammar of legitimate financial behavior by conceptualizing the history of transactions incurred by a user as a implication over time. The self-training methodology allows the model to detect the fraudulent activities as serious undergoing disparities of these acquired patterns. This methodology is effective as confirmed by the empirical findings. The hybrid model performed better than the known machine learning baselines and existing recurrent architectures and achieves F1-Score of 92.1% and recall of 91.5%. This recall rate is especially important, as it helps to stress the fact that the model is very strong in false negative reduction to give a greater assurance of security. The analysis of the qualitative information also indicated that the model is competent in detecting the presence of complex fraud typologies, including account takeovers and social engineering scams, that are usually difficult to detect by traditional systems. Having a low prediction latency and a controllable memory footprint, besides being accurate, the system is also operationally viable to be deployed to a high-throughput financial environment in real time. The article brings an additional adaptive, behavior-based security paradigm that may adapt to the new fraud strategies.

References

1. Carcillo, F., Le Borgne, Y. A., Caelen, O., & Bontempi, G. (2021). A survey of machine learning for financial fraud detection. IEEE Transactions on Neural Networks and Learning Systems, 32(5), 1888-1902.
2. Mienye, I. D., & Jere, N. (2024). Sequential deep learning frameworks for robust financial fraud detection: A comparative study of temporal and non-temporal models. International Journal of Engineering Development and Research, 12(4), 535–545.
3. Bello, A., & Others. (2023). Deep learning architectures for financial fraud detection. IEEE Access, 11, 12345–12356.
4. Cherif, A., & Others. (2023). Temporal pattern recognition in financial fraud using LSTM networks. Neural Computing and Applications, 35, 8901–8915.
5. Wang, S., & Li, B. (2022). Modeling user behavior sequences with RNN for fraud detection. Future Generation Computer Systems, 112, 123-134.
6. Lokanan, M. (2023). Momentum-based deep learning for mobile money fraud detection. Journal of Financial Crime, 30(2), 450–468.
7. Igwesi, I. (2023). Combating mobile money theft with CNN-BiLSTM and optimized SGD. Journal of Information Security, 14(3), 1–24.
8. Cheng, D., Zou, Y., Xiang, S., & Jiang, C. (2024). Graph neural networks for financial fraud detection: A review. arXiv preprint arXiv:2411.05815.
9. Kesharwani, A., & Shukla, P. (2024). FFDM - GNN: A financial fraud detection model using graph neural network. 2024 International Conference on Computing, Sciences and Communications (ICCSC), 1–6.
10. u, Q., Yin, Y., Zhou, S., Mu, H., & Hu, Z. (2025). Detecting financial fraud in listed companies via a CNN-transformer framework. Preprints.org.
11. Cui, Y., Han, X., Chen, J., Zhang, X., Yang, J., & Zhang, X. (2025). FraudGNN-RL: A graph neural network with reinforcement learning for adaptive financial fraud detection. IEEE Open Journal of the Computer Society.
12. Naim, B., Rees, B., & Liu, S. (2025). Supercharging Fraud Detection in Financial Services with Graph Neural Networks. NVIDIA Technical Blog.
13. Kasyap, H., Atmaca, U. I., & Maple, C. (2024). Privacy-preserving personalised federated learning financial fraud detection. IET Conference Proceedings, 2024(7), 1-6.
14. Rizvi, S., & Others. (2025). Secure and Transparent Banking: Explainable AI-Driven Federated Learning Model for Financial Fraud Detection. MDPI Journal of Risk and Financial Management, 18(4), 179.
15. Ali, R. (2026). Explainable AI framework for credit card fraud detection using XGBoost, deep neural

networks and SHAP-based interpretability. SSRN Electronic Journal.
16. Chagahi, M. H., Delfan, N., Dashtaki, S. M., Moshiri, B., & Piran, M. J. (2024). Explainable AI for fraud detection: An attention-based ensemble of CNNs, GNNs, and a confidence-driven gating mechanism. arXiv preprint arXiv:2410.09069.
17. IEEE Xplore. (2025). Explainable AI for Fraud Detection: Enhancing Transparency and Trust in Financial Decision-Making. 2024 International Conference on AI.
18. Zhao, Y., & Guan, H. (2025). Gated attention based generative adversarial networks for imbalanced credit card fraud detection. PeerJ Computer Science, 11, e2972.
19. Suginoo, M. (2024). Fraud Detection with Generative Adversarial Nets (GANs). Medium Data Science.
20. Eteng, S., Paul, M., & John, G. (2024). Self-Supervised Learning Approaches for Detecting Rare and Evolving Financial Fraud Scenarios. ResearchGate Publications.
21. Preprints.org. (2025). Self-Supervised Learning for Financial Statement Fraud Detection with Limited and Imbalanced Data. Preprints.org.
22. Shah, P., & Sharma, S. (2023). Adaptive fraud detection using ensemble machine learning. International Journal of Intelligent Systems, 38(4), 1120–1145.
23. Btoush, M. H., & Others. (2025). Hybrid machine learning and deep learning approaches for fraud detection. Journal of Cybersecurity and Privacy, 5(1), 112–129.
24. Khalid, R., & Others. (2024). Data augmentation techniques for financial fraud detection. Applied Sciences, 14(5), 2345.
25. Tiwari, S. (2025). Enhancing Financial Crime Detection through Data Science-Driven Transaction Monitoring. International Journal of Computing and Engineering, 7(13), 53-63.