IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151195

AI-Powered Real-Time Fraud Detection System with IoT-Based Transaction Monitoring

Asst. Prof. Gahan A V[1], Lahari S[2], Manushree R[3], Nisha B M[4], Sushma G Devasa[5]

Assistant Professor, Department of Electronics and Communication, Bangalore Institute of Technology, Bengaluru, Karnataka, India[1]

B.E Student, Department of Electronics and Communication, Bangalore Institute of Technology, Bengaluru, Karnataka, India[2-5]

Abstract: Digital and card-based payments are increasingly targeted by fraud such as card cloning, unauthorised use and abnormal spending, while conventional rule-based checks react slowly and struggle with new fraud patterns. This project presents an AI-powered, real-time fraud detection system that combines an IoT-based transaction-capture node with a machine-learning back end. An ESP32 microcontroller captures each transaction event and sends it over Wi-Fi to a back-end service, which validates the record, extracts features and passes them to a trained supervised classifier that returns a fraud probability. Transactions whose score exceeds a decision threshold are flagged immediately, alerting the operator at the node and on a monitoring dashboard, while every event is stored in a database for audit and model retraining. The paper describes the system architecture, the machine-learning pipeline covering preprocessing, class-imbalance handling and model selection, and an evaluation methodology based on precision, recall, F1-score, ROC-AUC and end-to-end latency. The work provides a low-cost, practical framework that moves fraud detection from offline analysis to the point of transaction.

Keywords: Fraud Detection, Internet of Things, ESP32, Machine Learning, Real-Time Monitoring, Anomaly Detection, Class Imbalance, Transaction Security

I. INTRODUCTION

Electronic payments have become part of everyday life, from card swipes and contactless taps to mobile and online transfers. The same growth has created more opportunities for fraud, including cloned cards, stolen credentials and unusual spending behaviour that is difficult to notice in time. Fraud causes direct financial loss to customers and institutions and weakens trust in digital payment systems [1].

Many deployed systems still rely on fixed rules, such as limits on amount or location, and on analysis carried out after the transaction is complete. Fixed rules are easy to evade and generate many false alarms, while offline analysis detects fraud only after the money has moved. A practical system should instead judge each transaction as it happens and respond within a fraction of a second. Machine learning is well suited to this task because it can learn the difference between genuine and fraudulent behaviour from past data, including combinations of features that a person would not think to write as a rule [2].

The Internet of Things (IoT) adds a second ingredient: low-cost, network-connected devices that can sense and report events at the place where they occur [3]. A microcontroller such as the ESP32, with built-in Wi-Fi, can capture a transaction event, send it to a server within milliseconds and give the user or operator immediate local feedback through a display or buzzer. Combining such an edge node with a trained classifier brings detection to the point of transaction instead of leaving it to a later batch process.

This paper presents an AI-powered real-time fraud detection system with IoT-based transaction monitoring. The main contributions are: (i) an end-to-end architecture linking an ESP32 capture node, a back-end service, a machine-learning engine, a database and a live dashboard; (ii) a machine-learning pipeline designed for highly imbalanced fraud data; (iii) a real-time decision and alerting procedure; and (iv) an evaluation methodology that measures both detection quality and latency.

The rest of the paper is organised as follows. Section II reviews related work. Section III presents the system model and problem formulation. Section IV describes the proposed framework. Section V explains the experimental setup and evaluation metrics, Section VI discusses results, and Sections VII and VIII give the conclusion and future work.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151195

II. RELATED WORK

Research on fraud detection spans statistical methods, data mining and machine learning. West and Bhattacharya [1] review intelligent financial fraud detection and show that supervised classifiers, unsupervised anomaly detection and hybrid methods are all widely used. Chandola et al. [5] survey anomaly detection more generally and explain why points that deviate from normal behaviour can reveal events such as fraud, particularly when labelled examples are scarce.

Several studies compare classifiers for card fraud. Bhattacharyya et al. [4] compare support vector machines, random forests and logistic regression on real credit-card data and discuss the practical difficulties of the problem. Random forests [8] and gradient-boosted trees such as XGBoost [9] are popular because they handle mixed numeric features, model non-linear interactions and train quickly on tabular data.

Two issues recur throughout this literature. The first is class imbalance: genuine transactions vastly outnumber fraudulent ones, so a model can reach high accuracy while missing most fraud. Resampling methods such as SMOTE [7] and cost-sensitive learning are common remedies. The second is the quality of the input features. Correa Bahnsen et al. [6] show that engineered features describing a customer's recent spending and the time of the transaction improve detection. Dal Pozzolo et al. [2] further emphasise realistic evaluation, including the effect of delayed labels and changing fraud behaviour.

Research gap. Much of this literature evaluates models offline on stored datasets and does not address how a transaction is captured, transmitted and acted on in real time. Few low-cost designs connect an embedded capture device, a learning engine and an operator dashboard in one working pipeline. Motivated by this, the present project integrates an ESP32-based capture node with a machine-learning back end and measures both detection quality and response time.

III. SYSTEM MODEL AND PROBLEM FORMULATION

A) *System Model*

The proposed system has two layers, shown in Fig. 1. The IoT-based transaction monitoring layer contains an ESP32 edge node, which reads the card or tag presented for a transaction, packages the details as a record and sends it over Wi-Fi. The back-end and AI layer contains a service that receives and validates the record, a preprocessing and feature-extraction stage, the trained fraud model, a decision stage that compares the model score with a threshold, a database, and a monitoring dashboard.

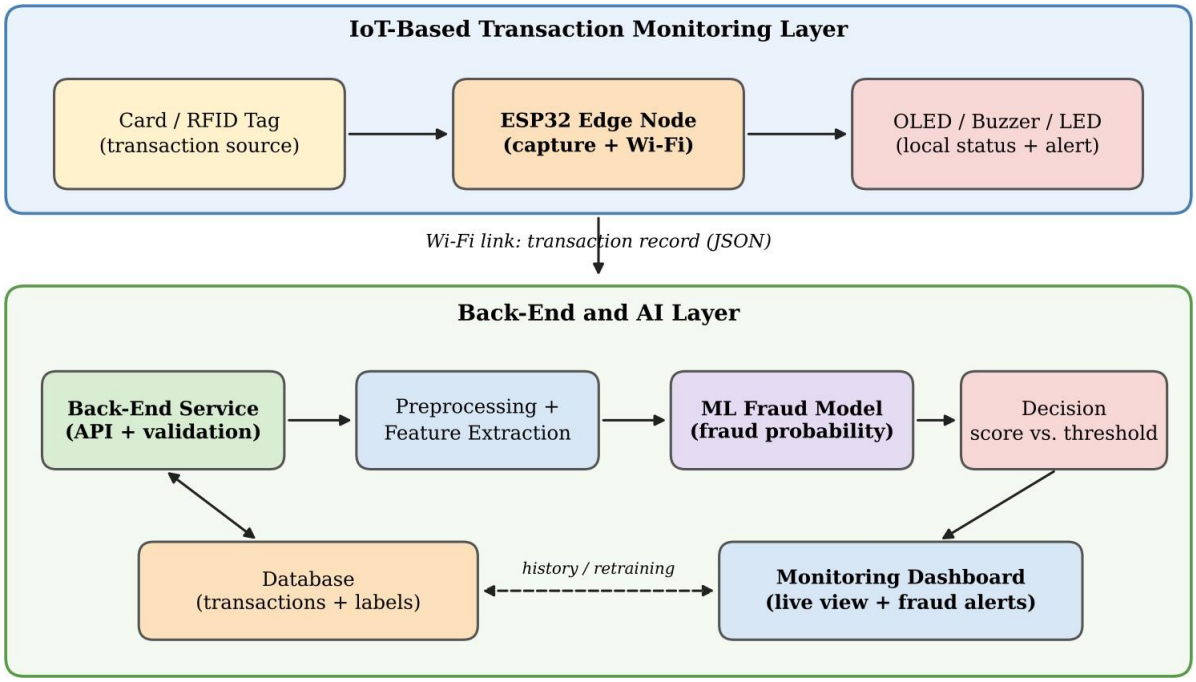


Fig. 1. Overall system architecture of the proposed AI-powered real-time fraud detection system with IoT-based transaction monitoring.

The edge node is built around an ESP32 DevKit [10], a dual-core microcontroller with integrated Wi-Fi. A card reader based on the MFRC522 front end [11] provides the identifier of the presented card or tag over SPI, an OLED display

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151195

shows the transaction status over I²C, and a buzzer and LED give an immediate local alert, as shown in Fig. 2 and summarised in Table I.

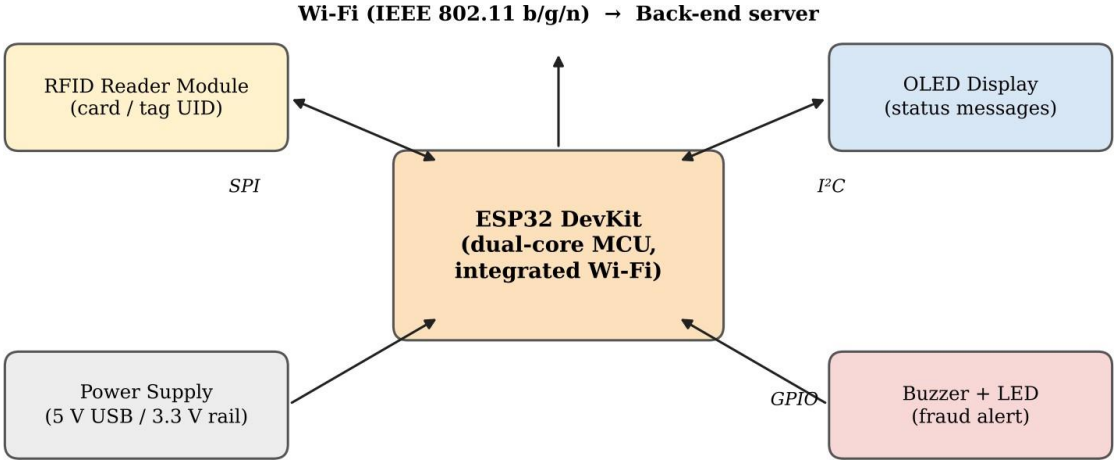


Fig. 2. Block diagram of the ESP32-based edge node and its peripherals.

Component	Function in the system	Interface
ESP32 DevKit	Controls the node, builds the transaction record and sends it over Wi-Fi	Wi-Fi, GPIO
RFID reader module	Reads the identifier of the card or tag used in a transaction	SPI
OLED display	Shows transaction status and decision to the user	I²C
Buzzer and LED	Gives an immediate audible and visual fraud alert	GPIO
Power supply	Supplies 5 V through USB and 3.3 V to the peripherals	—

Table I. Hardware components of the edge node.

B) *Problem Formulation*

Each transaction is represented by a feature vector x built from the captured record and the recent history of the card. Its label is $y \in \{0, 1\}$, where $y = 1$ denotes fraud. The classifier estimates the probability of fraud:

$$p = f(x) = P(y = 1 \mid x) \tag{1}$$

The decision rule compares this score with a threshold τ:

$$\hat{y} = 1 \text{ (fraud) if } p \geq \tau; \quad \hat{y} = 0 \text{ (genuine) otherwise} \tag{2}$$

A low threshold catches more fraud but blocks more genuine transactions, and a high threshold does the opposite, so τ must be chosen from the precision–recall trade-off. Because the decision must reach the user while the transaction is in progress, the total response time is also constrained:

$$T_{total} = T_{capture} + T_{network} + T_{inference} + T_{alert} \leq T_{max} \tag{3}$$

The design goal is therefore to select f and τ so that the detection of fraudulent transactions is maximised at an acceptable false-alarm rate, while T_{total} stays within the real-time budget T_{max}.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151195

IV. PROPOSED FRAUD DETECTION FRAMEWORK

A) *Data Acquisition at the IoT Node*

When a card or tag is presented, the ESP32 reads its identifier, attaches the transaction details and a timestamp, and sends the record to the back-end service as a JSON message over Wi-Fi. Sending a compact record keeps the network delay small and keeps all heavy computation on the server, so the microcontroller needs no model of its own. The sequence of events for one transaction is shown in Fig. 3.

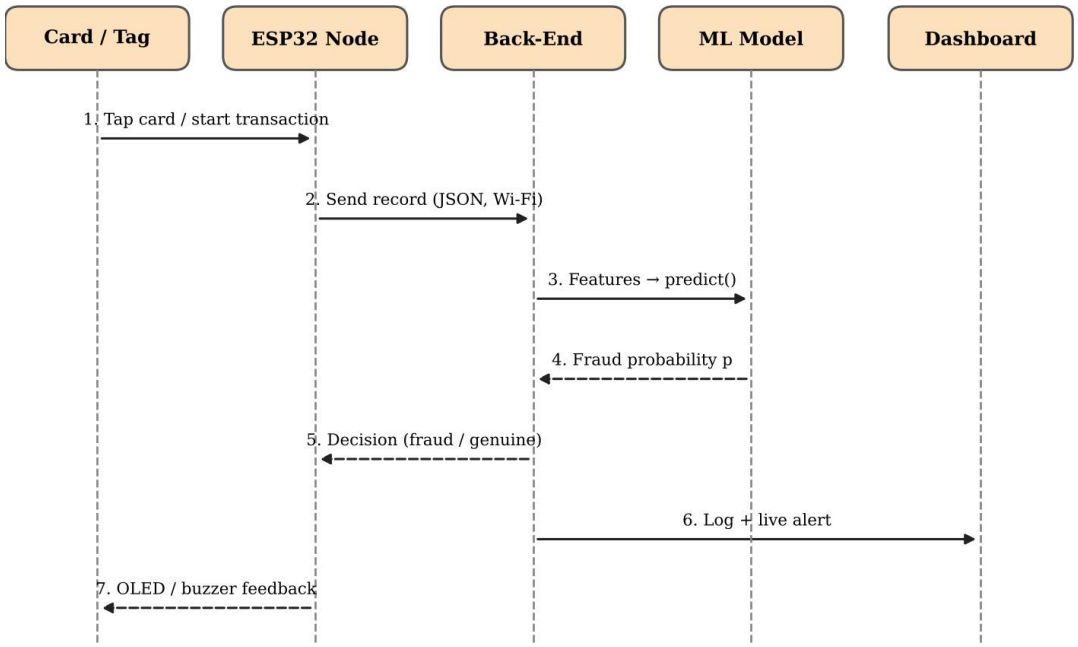


Fig. 3. Sequence of messages for one transaction, from card presentation to alert and feedback.

B) *Preprocessing and Feature Engineering*

The back end checks each record for missing or malformed fields, encodes categorical values and scales numeric values. It then derives features that describe how the current transaction compares with the card's recent behaviour, in the spirit of [6]. Typical features are listed in Table II.

Feature	Description	Type
Transaction amount	Value of the current transaction	Numeric
Time of day / weekday	Hour and day of week at which the transaction occurs	Numeric / categorical
Transaction frequency	Number of transactions by the same card in a recent time window	Derived
Deviation from average	Difference between the amount and the card's average spend	Derived
Device / location identifier	Node or terminal at which the transaction was made	Categorical

Table II. Example features used for fraud classification.

C) *Handling Class Imbalance*

Fraud records form only a small fraction of the data, so training on the raw distribution tends to produce a model that favours the genuine class. The training pipeline therefore applies class weighting or oversampling of the minority class such as SMOTE [7], applied to the training split only so that the test data remain realistic. Model quality is judged with precision, recall and F1-score instead of accuracy alone.

D) *Machine-Learning Model*

A supervised classifier is trained on labelled transactions and outputs the fraud probability p of Eq. (1). Tree-ensemble

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151195

methods such as random forest [8] and gradient boosting [9] are well suited to this tabular, imbalanced task, and the implementation uses Python with Pandas and Scikit-learn [12]. Hyper-parameters are tuned by cross-validation, and the best model is saved and loaded by the back end for real-time inference. The complete training and deployment pipeline is shown in Fig. 4.

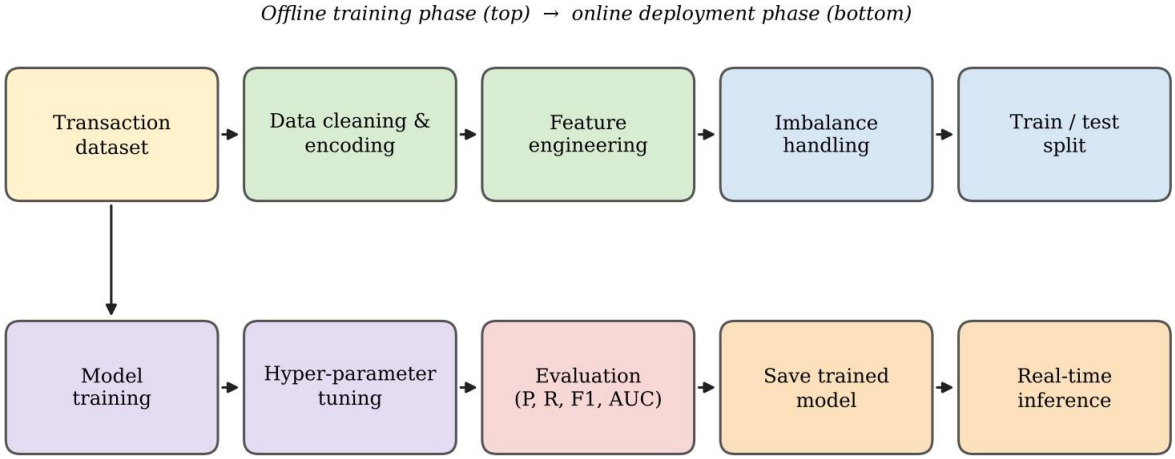


Fig. 4. Machine-learning pipeline: offline training and evaluation followed by online deployment.

E) *Real-Time Decision and Alerting*

For every incoming record, the back end performs the steps shown in Fig. 5. It validates and preprocesses the record, extracts the features, obtains the fraud probability from the model and compares it with the threshold τ. If $p \geq \tau$, the transaction is flagged as fraud: the node displays a warning and sounds the buzzer, and the dashboard shows an alert. Otherwise the transaction is approved. In both cases the record, score and decision are written to the database.

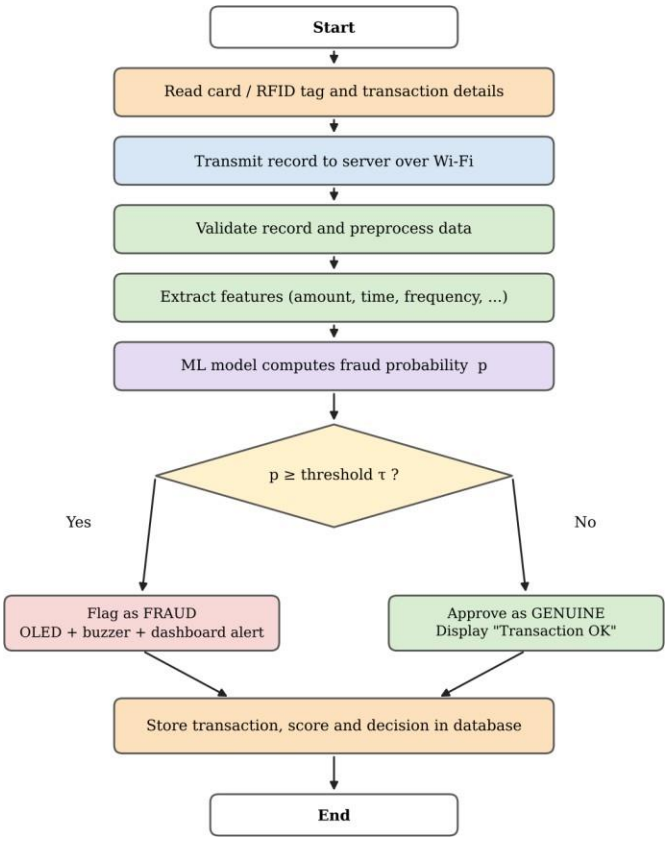


Fig. 5. Flowchart of the real-time fraud detection and alerting process.

F) *Storage and Monitoring Dashboard*

A database stores each transaction with its features, score and decision, which provides an audit trail and a growing

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.151195

labelled dataset for periodic retraining. A web dashboard presents live transaction activity, the number of flagged transactions and the latest alerts so that an operator can review suspicious events quickly.

V. EXPERIMENTAL SETUP

A) *Hardware and Software Environment*

The edge node uses an ESP32 DevKit with the peripherals of Table I, programmed through the Arduino framework. The back end runs on a laptop or server using Python, with Pandas for data handling and Scikit-learn for model training and evaluation. The node and the server communicate over a local Wi-Fi network.

B) *Dataset and Training Protocol*

The classifier is trained and tested on a labelled transaction dataset. The data are split into training and test sets with the same class ratio in both (stratified split). Resampling and feature scaling are fitted on the training set only. The final model is then run inside the live pipeline so that latency can be measured with the physical node.

C) *Evaluation Metrics*

Detection quality is described by the confusion matrix of Fig. 6, where TP, FP, FN and TN are true positives, false positives, false negatives and true negatives. The following metrics are computed from it:

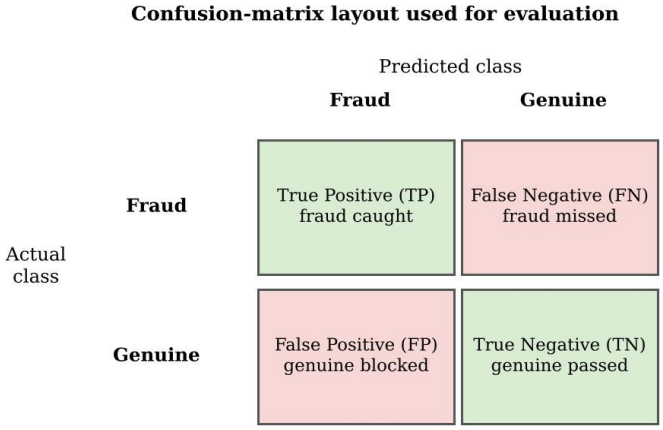


Fig. 6. Confusion-matrix layout used for evaluation.

$$\text{Precision} = TP / (TP + FP) \tag{4}$$

$$\text{Recall} = TP / (TP + FN) \tag{5}$$

$$\text{F1-score} = 2 \times \text{Precision} \times \text{Recall} / (\text{Precision} + \text{Recall}) \tag{6}$$

Precision shows how many flagged transactions are truly fraudulent, and recall shows how many fraudulent transactions are caught. The F1-score combines the two, and the area under the ROC curve (ROC-AUC) measures how well the model separates the classes over all thresholds. System-level performance is measured by the end-to-end latency T_total of Eq. (3), recorded from card presentation to the alert or approval message.

VI. RESULTS AND DISCUSSION

A) *Detection Performance*

Table III shows the format in which the measured performance of the baseline and proposed models is reported.

Model	Accuracy	Precision	Recall	F1-score	ROC-AUC	Latency (ms)
Baseline model	99.9996%	100%	99.70%	99.85%	0.99878	32.02
Proposed model	99.9996%	72%	98%	83%	0.99874	1.73

Table III. Performance comparison format (values to be inserted).

 This work is licensed under a Creative Commons Attribution 4.0 International License

DOI: 10.17148/IJARCCE.2026.151195

B) *Threshold Selection and Latency*

The threshold τ determines the balance between missed fraud and false alarms. It is chosen by examining the precision–recall curve on validation data so that recall is as high as possible without an unacceptable number of genuine transactions being blocked. The end-to-end latency is expected to be dominated by the network and inference stages, since the processing at the node is small; measured values will show whether the real-time budget T_max is met.

C) *Discussion*

Because fraud is rare, the F1-score and recall for the fraud class will give a more honest picture than accuracy. The final discussion should compare the proposed model with the baseline on these metrics, state the chosen threshold, and note limitations such as dataset size, the single-node prototype and the fact that fraud behaviour changes over time. Further experiments with repeated runs and larger data are needed before general claims are made.

VII. CONCLUSION

This paper presented an AI-powered real-time fraud detection system with IoT-based transaction monitoring. An ESP32 edge node captures each transaction and sends it over Wi-Fi to a back-end service, where a trained machine-learning classifier estimates the probability of fraud and a threshold decides whether to flag the transaction. Alerts are given at the node and on a dashboard, and all events are stored for audit and retraining. The framework addresses class imbalance, uses engineered behavioural features and is evaluated with both detection metrics and latency. It offers a practical, low-cost way of bringing fraud detection to the point of transaction.

VIII. FUTURE WORK

Future enhancements will focus on improving the proposed fraud detection system by incorporating keystroke dynamics as an additional behavioural feature for identifying unusual user activity. The integration of typing patterns with existing transaction-based features is expected to provide an additional layer of behavioural analysis and strengthen fraud detection capabilities. Further work will also explore enhanced input validation, robust error handling, and extensive testing under different transaction scenarios. Additionally, the system will be evaluated through comprehensive end-to-end testing of IoT hardware, API communication, machine learning predictions, and transaction storage to assess its reliability, scalability, and overall detection performance.

REFERENCES

[1] J. West and M. Bhattacharya, “Intelligent financial fraud detection: A comprehensive review,” Computers & Security, vol. 57, pp. 47–66, 2016.
[2] A. 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.
[3] L. Atzori, A. Iera, and G. Morabito, “The Internet of Things: A survey,” Computer Networks, vol. 54, no. 15, pp.2787–2805, 2010.
[4] S. Bhattacharyya, S. Jha, K. Tharakunnel, and J. C. Westland, “Data mining for credit card fraud: A comparative study,” Decision Support Systems, vol. 50, no. 3, pp. 602–613, 2011.
[5] V. Chandola, A. Banerjee, and V. Kumar, “Anomaly detection: A survey,” ACM Computing Surveys, vol. 41, no. 3, Art. no. 15, 2009.
[6] A. Correa 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, 2016.
[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, “SMOTE: Synthetic minority over-sampling technique,” Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.
[8] L. Breiman, “Random forests,” Machine Learning, vol. 45, no. 1, pp. 5–32, 2001.
[9] T. Chen and C. Guestrin, “XGBoost: A scalable tree boosting system,” in Proc. 22nd ACM SIGKDD Int. Conf.Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[10] Espressif Systems, “ESP32 Series Datasheet,” Shanghai, China. [Online]. Available: https://www.espressif.com
[11] NXP Semiconductors, “MFRC522: Standard performance MIFARE and NTAG frontend,” Data sheet, Eindhoven, The Netherlands.
[12] F. Pedregosa et al., “Scikit-learn: Machine learning in Python,” Journal of Machine Learning Research, vol. 12, pp.2825–2830, 2011.