International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

BANKING FRAUD DETECTION USING ARTIFICIAL NEURAL NETWORK (ANN)

Yeddula Naresh Kumar[1], Karanam Pavan Kumar [2]
[1] *M. Tech Student, Department of Computer Science and Engineering*
[2] *Associate Professor, Department of Computer Science and Engineering*
Bharatiya Engineering Science and Technology Innovation University (BESTIU)
Gownivaripalli, Gorantla, Satya Sai (Dt) – 515231, Andhra Pradesh, India

Abstract Banking fraud causes direct financial loss, weakens customer trust and exposes institutions to regulatory risk, and it is hard to detect because fraudulent transactions are rare and fraud tactics keep changing. Rule-based controls and shallow classifiers struggle to capture the non-linear interactions among transaction attributes. This paper presents an end-to-end fraud detection system built around a feed-forward Artificial Neural Network (ANN) implemented with the MLPClassifier of scikit-learn. The network has three hidden layers (64–32–16 neurons, ReLU activation) and a logistic output unit that estimates the probability that a transaction is fraudulent. Raw transaction records (over one million rows) are joined with non-sensitive card metadata, enriched with temporal, monetary and card-profile features, standardised, and drawn as a stratified sample for training with Adam optimisation and early stopping; card numbers and CVV values are excluded. The trained model is wrapped in a batch pipeline that streams large CSV files in chunks and produces risk-band reports, and in a Flask web console for single-transaction scoring, dataset upload, retraining and report download. On an evaluation set of 31,000 transactions the ANN reaches a precision of 79.75%, a recall of 83.62%, an F1-score of 81.64%, a specificity of 98.58% and an AUC-ROC of 0.9821, and it scores higher than Logistic Regression, Random Forest, SVM, XGBoost and Naive Bayes baselines on accuracy, precision, recall, F1-score and AUC-ROC, at the cost of the longest training time. The paper also discusses limitations, in particular that the fraud label is derived rather than bank-confirmed and that results come from a single data split, so the findings should be confirmed on verified bank data before deployment.

Keywords Banking fraud detection · Artificial neural network · Multilayer perceptron · Feature engineering · Risk scoring · Class imbalance · Flask · Financial security

I. INTRODUCTION

Internet banking, mobile wallets, instant payment rails and card-not-present commerce have made financial services faster and more convenient, but they have also enlarged the surface that fraudsters can attack. Stolen-card use, account takeover, phishing-driven transfers, identity theft and money-mule activity all end in direct losses for customers or banks, and they additionally damage reputation and attract regulatory attention. A bank that processes millions of transactions a day cannot verify them manually; it needs an automated system that scores every transaction, ranks the riskiest ones for investigation and explains its output to analysts and compliance teams.

Conventional fraud controls are built from hand-written rules such as amount limits, blacklists and velocity checks. Rules are transparent but brittle: they must be revised by hand, they miss patterns nobody has written a rule for, and they raise many false alarms when genuine customers behave unusually. Classical machine learning improves on rules, yet linear models cannot represent complicated feature interactions and single decision trees are unstable. Artificial Neural Networks (ANNs) are universal function approximators [1] that learn layered non-linear representations from data [2], which makes them well suited to modelling the subtle combinations of amount, time, merchant and card behaviour that separate fraud from normal activity.

Turning an ANN into a working fraud detection tool raises several practical problems. Fraud is a rare event, so accuracy alone is a misleading measure; real banking exports often come without a confirmed fraud flag, so a training label has to be constructed; card numbers and security codes must be kept out of the model for

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

privacy reasons; and the result must be usable by analysts who work with spreadsheets and dashboards rather than with Python notebooks. This paper addresses these problems in a single system.

The main contributions of this paper are:

(1) An ANN-based fraud scoring engine, implemented with scikit-learn's MLPClassifier and trained on a stratified sample of more than one million banking transactions joined with card metadata, using privacy-preserving temporal, monetary and card-profile features.

(2) A scalable batch pipeline that streams large transaction CSV files in chunks, scores every row, assigns it to a risk band and writes Markdown, JSON and CSV risk reports that list the top suspicious transactions.

(3) A Flask web console for single-transaction scoring, dataset upload with automatic analysis, a portfolio risk dashboard, model retraining and report download.

(4) An evaluation with precision, recall, F1-score, specificity, AUC-ROC and confusion-matrix analysis against five baseline classifiers, together with an open discussion of limitations.

The rest of the paper is organised as follows. Section 2 reviews related work. Section 3 describes the data, the ANN, the training procedure and the software architecture. Section 4 reports the experimental results. Section 5 discusses the findings and limitations, and Section 6 concludes the paper and outlines future work.

II. RELATED WORK

2.1 Rule-based and classical machine learning approaches

Early banking systems relied on predefined rules: transaction limits, blacklisted accounts and unusual spending thresholds. Such rules are easy to implement but cannot detect unseen patterns and need continual manual maintenance. Surveys of data-mining techniques for financial fraud, such as that of Ngai et al. [3], show that classification, clustering and outlier detection have all been applied to credit-card, insurance and corporate fraud. Bhattacharyya et al. [4] compared logistic regression, support vector machines and random forests on real credit-card data and stressed the importance of the evaluation metric when fraud is rare. Ensemble methods such as Random Forests [5] remain strong baselines because they handle mixed feature types with little tuning.

2.2 Neural and deep learning approaches

Neural networks trained by back-propagation [6] have long been used for transaction scoring. Jurgovsky et al. [7] treated the transaction history of a card holder as a sequence and showed that an LSTM-based classifier can improve on a random-forest baseline in card-present scenarios, and Carcillo et al. [8] combined unsupervised outlier scores with supervised classifiers on large-scale card data. Deeper and sequential models show the potential of deep learning but add training and deployment complexity; a compact feed-forward ANN remains attractive when low latency, simple maintenance and straightforward integration into a web application are priorities.

2.3 Class imbalance and evaluation

Class imbalance is the central practical difficulty. Chawla et al. [9] proposed SMOTE, which creates synthetic minority samples by interpolation, and Fiore et al. [11] used generative adversarial networks to create additional fraudulent examples. Dal Pozzolo et al. [10] showed that rebalancing the training data alters the predicted probabilities, so the decision threshold must be considered together with the sampling strategy. Saito and Rehmsmeier [12] showed that precision–recall measures are more informative than ROC curves on imbalanced data. These findings motivate the use of precision, recall, F1-score and a confusion matrix, in addition to AUC-ROC, in this paper.

2.4 Summary and research gap

Table 1 summarises the studies discussed above. Most published work evaluates models on pre-labelled benchmark data and stops at the classifier. Less attention is paid to the surrounding engineering: building features from raw transaction and card tables without using sensitive fields, scoring very large files in bounded memory, grouping scores into risk bands that analysts can act on, and exposing the model through a web interface. This paper addresses these gaps with a compact ANN embedded in a complete, deployable pipeline.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Table 1 Summary of related studies

Study	Technique	Focus	Observation relevant here
Ngai et al. [3]	Survey of data-mining methods	Financial fraud	Broad coverage; no single best method
Bhattacharyya et al. [4]	LR, SVM, Random Forest	Credit-card fraud	Metric choice matters for rare fraud
Jurgovsky et al. [7]	LSTM sequence classifier	Card-present fraud	History helps; higher complexity
Carcillo et al. [8]	Unsupervised + supervised	Large-scale card data	Outlier scores add useful features
Chawla et al. [9]	SMOTE	Imbalanced learning	Interpolated minority samples
Dal Pozzolo et al. [10]	Undersampling analysis	Unbalanced classification	Resampling shifts probabilities
Fiore et al. [11]	GAN-based augmentation	Card fraud	Synthetic fraud samples can help
Saito and Rehmsmeier [12]	PR vs ROC analysis	Imbalanced evaluation	PR curves are more informative

III. PROPOSED METHODOLOGY

3.1 Overview of the framework

The proposed framework, shown in Fig. 1, takes banking transactions through eight stages: data acquisition, merging and feature engineering, construction of a fraud label, stratified partitioning, standardisation, ANN training, threshold-based evaluation, and risk scoring with report generation. The trained model outputs a fraud probability for each transaction. A decision rule and a set of risk bands convert this probability into an action that an analyst can take: approve, review or block. All preprocessing steps are stored together with the model, so that a transaction scored from the web console or from a batch file passes through exactly the same transformations that were applied during training.

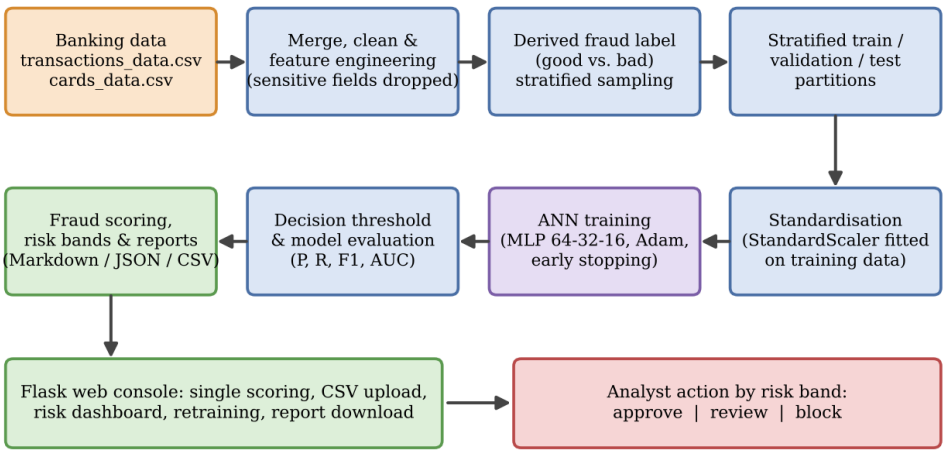


Fig. 1 Proposed banking fraud detection framework using an ANN

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

3.2 Datasets and fraud label

The system uses two CSV files. The file *transactions_data.csv* contains more than one million banking transactions, and *cards_data.csv* contains card-level metadata. Table 2 lists the attribute groups. Sensitive fields, namely card numbers and CVV values, are never passed to the model. The source files do not carry a confirmed fraud flag, so a binary target y is derived from the transaction records, with $y = 1$ for a bad (fraud-like) transaction and $y = 0$ for a good one. The ANN therefore learns to reproduce this derived label; the consequences are discussed in Section 5.1.

Table 2 Attribute groups of the banking datasets

Attribute group	Examples	Role in the system
Transaction details	Transaction ID, timestamp, amount, error type	Source of temporal and monetary features
Merchant and location	Merchant name, merchant category, city, state	Source of merchant and geographic features
Customer and card keys	Customer ID, card ID	Join keys between the two files
Card metadata	Card type and other non-sensitive card attributes	Card-profile features
Sensitive card fields	Card number, CVV	Excluded from the model
Target	Derived good / bad (fraud-like) label	Training and evaluation target

The modelling sample drawn from the full data contains 297,484 records, of which 284,315 (95.6%) are good and 13,169 (4.4%) are bad, as shown in Fig. 2. The classes are therefore strongly imbalanced, and a classifier that labelled every record as good would already be right about 96% of the time on this sample. For that reason stratified sampling is used throughout, so that every partition keeps the class ratio of the sample, and the evaluation relies on precision, recall and F1-score rather than on accuracy.

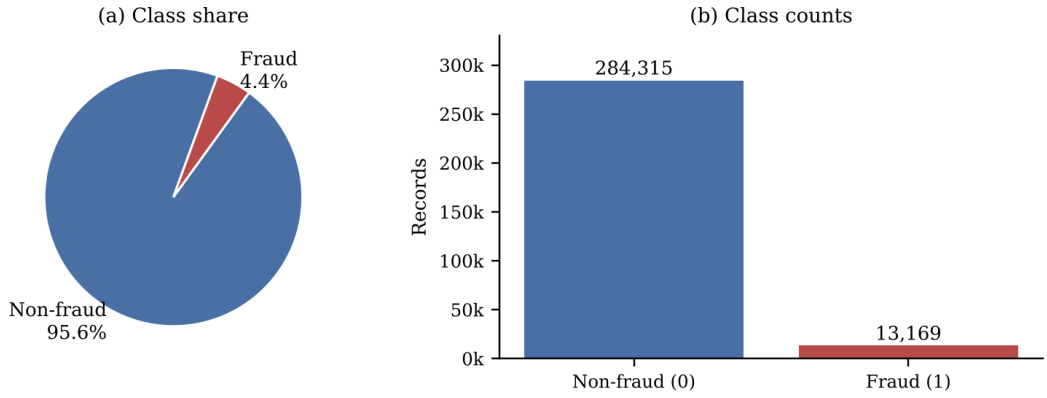


Fig. 2 Class distribution of the modelling sample: (a) share and (b) record counts

3.3 Preprocessing and feature engineering

Transactions are first joined with card metadata on the customer and card keys. Missing and duplicate records are removed, and categorical attributes are encoded numerically. Features are then derived in three groups: temporal features from the timestamp (for example the hour of the day and the day of the week), monetary features from the transaction amount, and card-profile features from the card metadata and from usage patterns such as the transaction frequency of a card. Because gradient-based training is sensitive to feature scale, every feature is standardised to zero mean and unit variance. The mean μ and standard deviation σ of each feature

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

are estimated on the training partition only and applied unchanged to the validation and evaluation data, which prevents information leakage:

$$x' = \frac{x - \mu}{\sigma} \quad (1)$$

Let d denote the number of features after encoding and standardisation, so that each transaction is a vector $\mathbf{x} \in \mathbb{R}^d$. The feature construction and the scaler are implemented as one scikit-learn pipeline object that is saved with the model.

3.4 ANN architecture

The classifier is a fully connected feed-forward network (Fig. 3) with an input layer of d units, three hidden layers of 64, 32 and 16 neurons, and one output neuron. For layer l the computation is

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)} \quad (2)$$

$$a^{(l)} = f\left(z^{(l)}\right) \quad (3)$$

where $W^{(l)}$ and $b^{(l)}$ are the weight matrix and bias vector of layer l, $a^{(0)} = \mathbf{x}$ is the input and f is the activation function. The hidden layers use the Rectified Linear Unit (ReLU) [13], which avoids vanishing gradients and is cheap to compute:

$$f(z) = max(0, z) \quad (4)$$

The output neuron uses the logistic sigmoid, so that its value can be read as the probability that the transaction is fraudulent:

$$\hat{y} = \sigma(z) = \frac{1}{1 + e^{-z}} \quad (5)$$

The number of trainable parameters follows from the layer sizes (weights plus biases of the four weight layers):

$$N_p = 64(d + 1) + 32 \cdot 65 + 16 \cdot 33 + 17 = 64d + 2689 \quad (6)$$

For example, a 30-feature input would give 4,609 parameters. The network is therefore small enough for millisecond-level scoring on ordinary hardware.

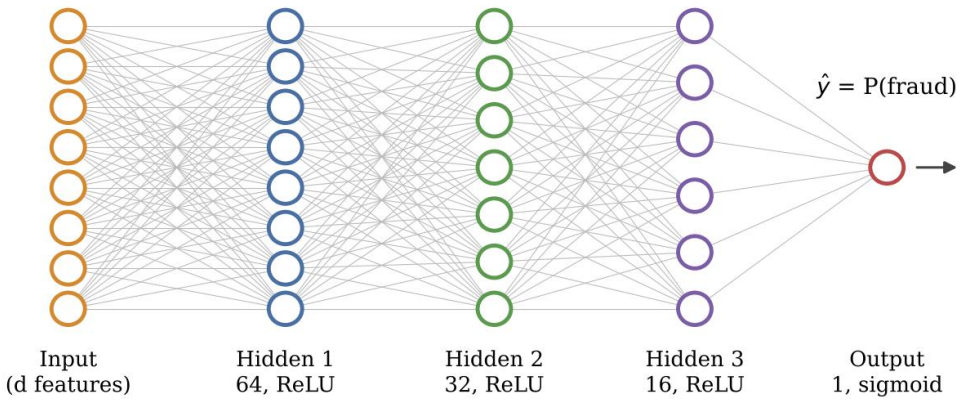


Fig. 3 Architecture of the ANN used for fraud detection (d–64–32–16–1)

3.5 Training procedure

The network is trained by mini-batch back-propagation [6] to minimise the binary cross-entropy (log) loss with an L2 weight penalty over N training samples:

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \, log \, \hat{y}_i + (1 - y_i) \, log(1 - \hat{y}_i)] + \frac{\alpha}{2}\sum_l \left\|W^{(l)}\right\|^2 \quad (7)$$

Parameters θ are updated with the Adam optimiser [14]:

$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m_t}}{\sqrt{\hat{v_t}} + \varepsilon} \quad (8)$$

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

where η is the learning rate, and $\hat{m}_t$ and $\hat{v}_t$ are the bias-corrected first and second moment estimates of the gradient. A fraction of the training data is held out as a validation set; training stops when the validation score no longer improves and the best weights are kept (early stopping), which limits over-fitting. Table 3 lists the hyper-parameters.

Table 3 Hyper-parameters of the proposed ANN

Hyper-parameter	Value
Implementation	scikit-learn MLPClassifier
Hidden layers / neurons	3 layers: 64, 32, 16
Hidden activation / output	ReLU / logistic sigmoid
Loss function	Binary cross-entropy (log loss) with L2 penalty
Optimiser	Adam (η = 0.001)
Batch size	256
Validation split for early stopping	20% of the training data
Maximum epochs	100, with early stopping
Input scaling	StandardScaler fitted on training data only

3.6 Decision rule and risk bands

A transaction is classified as fraudulent when its predicted probability reaches a threshold τ:

$$\text{class(x)} = \begin{cases} 1\,(fraud) & if\ \hat{y} \geq \tau \\ 0\,(legitimate) & otherwise \end{cases} \quad (9)$$

A lower τ catches more fraud but sends more genuine customers to manual review; a higher τ does the opposite. The threshold is therefore a configurable operating point that can be tuned on validation data, for example to maximise the F1-score or to meet a minimum precision. For reporting, the probability is also mapped to one of three risk bands with two configurable cut-offs $\tau_1 < \tau_2$:

$$\text{band(x)} = \begin{cases} Low & if\ \hat{y} < \tau_1 \\ Medium & if\ \tau_1 \leq \hat{y} < \tau_2 \\ High & if\ \hat{y} \geq \tau_2 \end{cases} \quad (10)$$

The risk-band distribution is the headline table of each batch report and of the portfolio dashboard.

3.7 Software architecture and reporting pipeline

The system follows the four-layer structure of Fig. 4. The presentation layer is a browser-based console, the application layer is a Flask service, the model layer holds the feature pipeline and the ANN, and the data layer holds the CSV files, the saved model and the generated reports. Table 4 maps the layers to the source modules.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

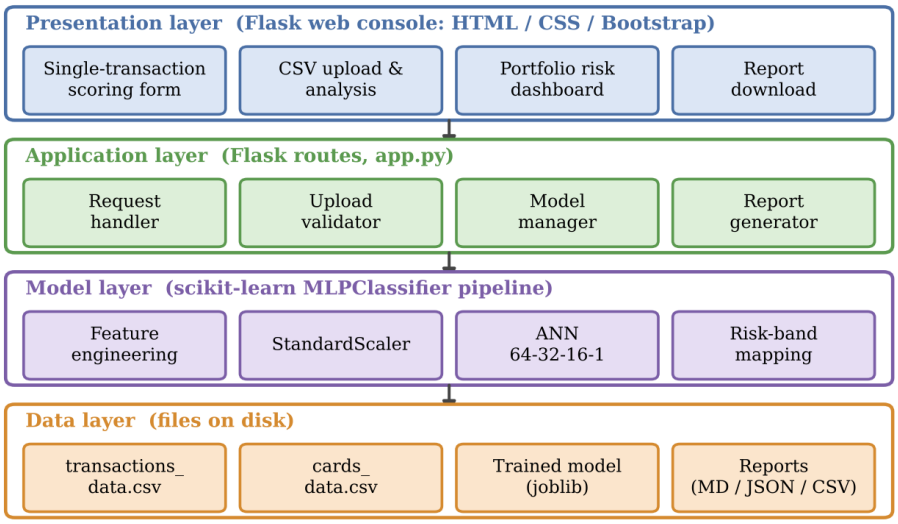


Fig. 4 Layered architecture of the fraud detection platform

Table 4 Software modules of the platform

Module	Responsibility
fraud_model.py	Feature engineering and the scikit-learn pipeline shared by all components
train_fraud_ann.py	Stratified sampling, ANN training and model saving
generate_fraud_report.py	Chunked scoring of large CSV files; risk-band tables and top suspicious transactions as Markdown, JSON and CSV
predict_transaction.py	Scoring of a single transaction
app.py	Flask console: single scoring, CSV upload with automatic analysis, portfolio dashboard, retraining, downloads

Large files are processed in chunks so that memory use stays bounded regardless of file size: each chunk is read, passed through the saved pipeline, scored and added to running risk-band counts and to a list of the highest-scoring transactions. The final report therefore contains the risk-band distribution and the top suspicious transactions for the whole file, without loading it at once.

3.8 Algorithm

Algorithm 1 ANN-based banking fraud detection

Input: transactions_data.csv, cards_data.csv; threshold τ; risk cut-offs $\tau_1 < \tau_2$
Output: trained ANN, fraud probabilities, risk bands and reports
1. Join transactions with non-sensitive card metadata; drop sensitive fields.
2. Remove missing values and duplicates; derive temporal, monetary and card-profile features.
3. Derive the binary label y; draw a stratified sample and split it into training, validation and test sets.
4. Estimate μ and σ on the training set; standardise all sets using Eq. (1).
5. Train the 64–32–16 ANN with Adam (Eq. 8) on the loss of Eq. (7); stop early on the validation set.
6. Select the threshold τ on validation data and store the pipeline and model.
7. For every chunk of an input file: compute $\hat{y}$ (Eq. 5), apply Eq. (9) and Eq. (10), update the risk-band counts and the top-k list.
8. Evaluate on the test set with Eqs. (11)–(15) and write the Markdown, JSON and CSV reports.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

3.9 Evaluation metrics

Let TP, TN, FP and FN denote true positives (frauds caught), true negatives, false positives (genuine transactions flagged) and false negatives (missed frauds). Because fraud is rare, accuracy alone is misleading, so the following metrics are reported as well:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{11}$$

$$\text{Precision} = \frac{TP}{TP + FP}, \quad \text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$\text{F1} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

$$\text{Specificity} = \frac{TN}{TN + FP}, \quad \text{FPR} = 1 - \text{Specificity} \tag{14}$$

$$\text{MCC} = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{15}$$

AUC-ROC is the area under the curve of true-positive rate against false-positive rate over all thresholds [15]. The Matthews correlation coefficient (MCC) summarises the whole confusion matrix in one number and remains informative when the classes are imbalanced.

IV. EXPERIMENTAL RESULTS

4.1 Experimental setup and datasets

The system was implemented in Python with scikit-learn [16], pandas, NumPy, joblib and Flask on a standard desktop computer; no GPU was used. Table 5 summarises the data used for modelling and for the final evaluation. The baselines are Logistic Regression, Random Forest, a Support Vector Machine (SVM), XGBoost and Naive Bayes, all evaluated on the same test data as the ANN.

Table 5 Datasets used in the experiments

Data	Records	Fraud-labelled	Fraud share
Modelling sample (stratified)	297,484	13,169	4.43%
Evaluation set (confusion matrix)	31,000	1,941	6.26%

4.2 Training dynamics

Table 6 and Fig. 5 show the loss and accuracy of the network at the epochs reported during training. Both losses fall steadily and both accuracies rise and level off. At the final epoch the training accuracy is 0.9951 and the validation accuracy 0.9654, while the training loss is 0.0128 and the validation loss 0.0876. The gap of about three percentage points in accuracy and a validation loss several times larger than the training loss point to mild over-fitting, which the L2 penalty and early stopping limit but do not remove.

Table 6 Training and validation behaviour at the reported epochs

Epoch	Training accuracy	Validation accuracy	Training loss	Validation loss
1	0.5671	0.5423	1.4052	1.3784
10	0.8423	0.8036	0.4126	0.5213
20	0.9136	0.8934	0.2218	0.2687
50	0.9752	0.9482	0.0421	0.1093
100	0.9951	0.9654	0.0128	0.0876

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

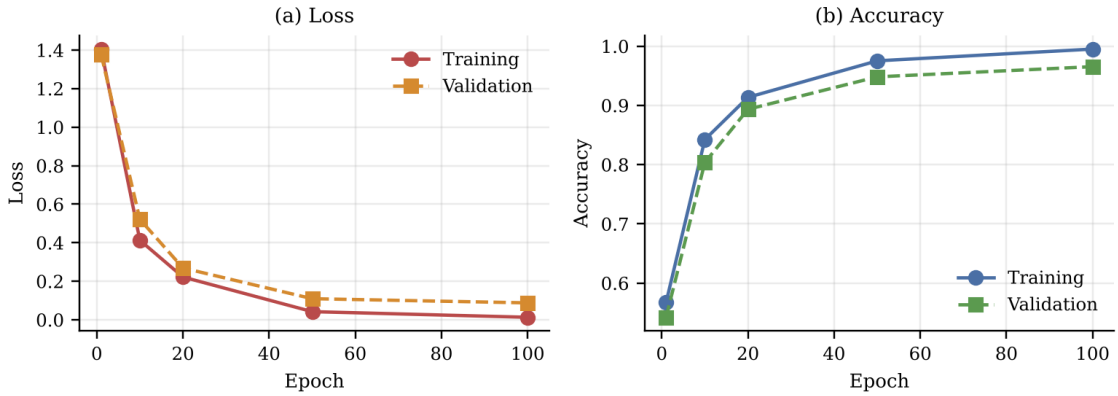


Fig. 5 Training and validation loss (a) and accuracy (b) at the reported epochs

4.3 Classification performance

Fig. 6 shows the confusion matrix of the ANN on the evaluation set of 31,000 transactions, of which 1,941 are fraud-labelled and 29,059 are good. The model detects 1,623 of the 1,941 fraud-labelled transactions, misses 318, and flags 412 of the 29,059 good transactions. Table 7 lists the metrics computed from this matrix with Eqs. (11)–(15), together with the AUC-ROC of 0.9821.

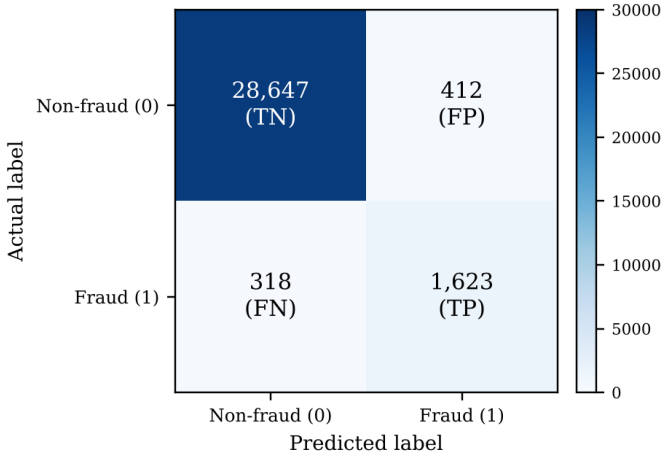


Fig. 6 Confusion matrix of the proposed ANN on the evaluation set

Table 7 Performance of the proposed ANN on the evaluation set

Metric	Value	Computed from
Accuracy	97.65%	(1,623 + 28,647) / 31,000
Precision	79.75%	1,623 / (1,623 + 412)
Recall (sensitivity)	83.62%	1,623 / (1,623 + 318)
F1-score	81.64%	Eq. (13)
Specificity	98.58%	28,647 / (28,647 + 412)
False-positive rate	1.42%	412 / 29,059
False-negative rate	16.38%	318 / 1,941
Matthews correlation coefficient	0.804	Eq. (15)

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

Metric	Value	Computed from
AUC-ROC	0.9821	Reported by the evaluation

The ANN catches about five of every six fraud-labelled transactions while raising an alert on only 1.42% of good ones, and about four of every five alerts are true fraud. The high specificity matters operationally: with more than 29,000 good transactions in the evaluation set, even a small false-positive rate produces hundreds of unnecessary reviews. The accuracy of 97.65% should be read with care, since a classifier that approved every transaction would score 93.74% on the same set; the MCC of 0.804 and the F1-score are more telling.

4.4 Comparison with baseline models

Table 8 and Fig. 7 compare the ANN with the five baselines on the same test data. The ANN obtains the highest accuracy, precision, recall, F1-score and AUC-ROC. Relative to XGBoost, the strongest baseline, it improves precision by 2.40, recall by 3.38 and F1-score by 2.88 percentage points, and AUC-ROC from 0.9723 to 0.9821. Random Forest and Logistic Regression follow, while Naive Bayes is clearly the weakest (F1-score 63.23%). The ANN is also the slowest model to train (142.35 s), roughly five times slower than Logistic Regression, which is the price of its higher detection quality.

Table 8 Test-set performance of the ANN and baseline models

Model	Acc. (%)	Precision (%)	Recall (%)	F1 (%)	AUC-ROC	Training time (s)
ANN (proposed)	**97.65**	**79.75**	**83.62**	**81.64**	**0.9821**	142.35
Logistic Regression	95.23	72.18	77.32	74.67	0.9483	28.47
Random Forest	96.81	76.35	79.65	77.96	0.9657	93.62
SVM	94.37	68.94	75.21	72.01	0.9364	37.88
XGBoost	97.12	77.35	80.24	78.76	0.9723	118.76
Naive Bayes	90.18	60.42	66.33	63.23	0.8932	18.63

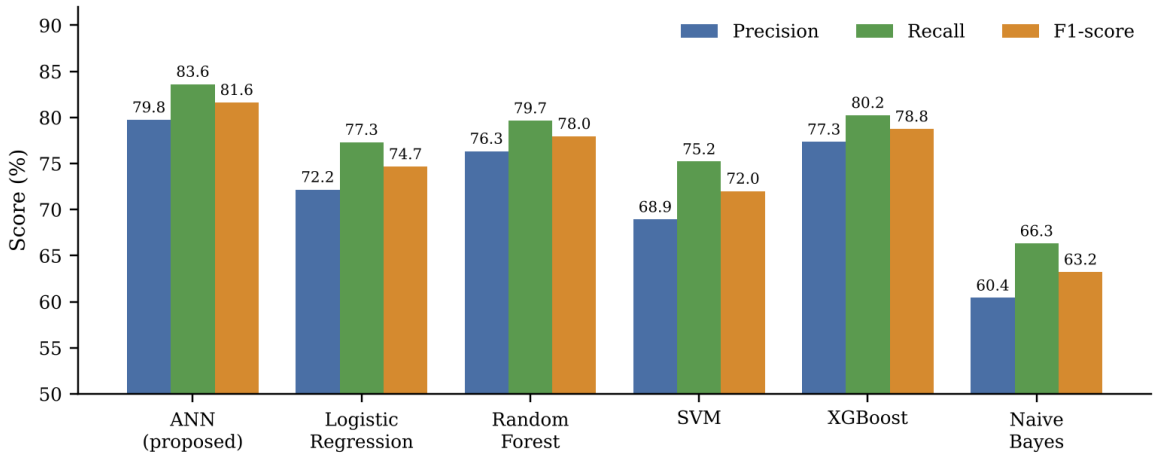


Fig. 7 Precision, recall and F1-score of the ANN and the baseline models

The margins over XGBoost and Random Forest are modest and come from a single data split without confidence intervals, so they show that the ANN is competitive and consistently ahead in this experiment, not that it is better than every tree ensemble in every setting.

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

4.5 Batch reports and web console

For every uploaded or supplied CSV file, the reporting pipeline scores all rows and produces a risk report in Markdown, JSON and CSV form. The report contains the distribution of transactions over the risk bands and a ranked list of the top suspicious transactions, so that analysts can begin an investigation with the highest-risk cases. The Flask console exposes the same functions interactively: an analyst can score a single transaction entered in a form, upload a dataset for automatic analysis, inspect the portfolio risk dashboard, trigger retraining of the model and download the generated reports.

V. DISCUSSION

The results show that a compact feed-forward ANN is an effective fraud classifier for this data. It leads all five baselines on accuracy, precision, recall, F1-score and AUC-ROC, keeps the false-positive rate at 1.42%, and has a parameter count of only $64d + 2{,}689$, so scoring a transaction costs a few thousand multiply–accumulate operations. In deployment the latency is therefore likely to be dominated by data retrieval and feature construction rather than by the model.

Three practical lessons emerge. First, accuracy is almost uninformative on imbalanced data, so precision, recall, F1-score and MCC should be reported. Second, privacy can be respected without losing performance: the model never sees card numbers or CVV values, yet it still reaches an F1-score of 81.64% from temporal, monetary and card-profile features. Third, a probability output is more useful than a hard label: risk bands let a bank approve low-risk transactions automatically, route medium-risk ones to step-up authentication or manual review, and block high-risk ones, with the cut-offs chosen according to the relative cost of a missed fraud and of an inconvenienced customer. Analyst decisions on reviewed cases can be fed back as new labels to retrain the model periodically, which is supported by the retraining control of the web console.

5.1 Limitations

(1) **Derived labels.** The fraud label is derived from the transaction records and not confirmed by investigators, so the reported metrics measure how well the ANN reproduces this label and may differ from performance against verified fraud.

(2) **Single split.** Results come from one partition of the data without cross-validation or confidence intervals, so small differences between the strongest models should not be over-interpreted. A time-based split would better reflect production use.

(3) **Limited tuning and analysis.** Baseline hyper-parameters were not exhaustively tuned, probability calibration and threshold sensitivity were not studied in detail, and precision–recall curves were not reported.

(4) **Static, unexplained scoring.** The model scores each transaction independently and does not use the customer's full history, and no explanation technique such as SHAP is applied, although analysts and regulators often require reasons for a decision.

(5) **Cost and drift.** The ANN is the slowest model to train, and periodic retraining is needed as fraud patterns drift.

VI. CONCLUSION AND FUTURE WORK

This paper presented an ANN-based banking fraud detection system that combines privacy-preserving feature engineering on transaction and card data, a 64–32–16 ReLU network with a logistic output trained with Adam and early stopping, a chunked batch pipeline that produces risk-band reports, and a Flask web console for analysts. On an evaluation set of 31,000 transactions the ANN reached a precision of 79.75%, a recall of 83.62%, an F1-score of 81.64%, a specificity of 98.58% and an AUC-ROC of 0.9821, and it outperformed Logistic Regression, Random Forest, SVM, XGBoost and Naive Bayes on every quality metric reported. The study also shows the importance of judging a fraud model by precision, recall and MCC rather than accuracy, and of presenting its output as risk bands that match analysts' workflows.

Future work will (i) validate the system on bank-confirmed fraud labels and on time-based splits, (ii) add class-weighting or SMOTE-style rebalancing [9] and threshold calibration studies, (iii) add customer-level sequence

International Journal of
DATA SCIENCE AND IOT MANAGEMENT SYSTEM
Peer Reviewed, Referred & Indexed Journal
ISSN: 3068-272X www.ijdim.com Original Research Paper

features and compare with gradient-boosted trees and autoencoder-based anomaly detectors, (iv) provide SHAP-based explanations for each alert, (v) connect the scoring service to real-time transaction streams and cloud deployment, and (vi) study concept drift with online or incremental retraining.

REFERENCES

1. K. Hornik, M. Stinchcombe, H. White, Multilayer feedforward networks are universal approximators. Neural Netw. 2(5), 359–366 (1989)
2. I. Goodfellow, Y. Bengio, A. Courville, Deep Learning (MIT Press, Cambridge, 2016)
3. E.W.T. Ngai, Y. Hu, Y.H. Wong, Y. Chen, X. Sun, The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. Decis. Support Syst. 50(3), 559–569 (2011)
4. S. Bhattacharyya, S. Jha, K. Tharakunnel, J.C. Westland, Data mining for credit card fraud: a comparative study. Decis. Support Syst. 50(3), 602–613 (2011)
5. L. Breiman, Random forests. Mach. Learn. 45(1), 5–32 (2001)
6. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
7. J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, O. Caelen, Sequence classification for credit-card fraud detection. Expert Syst. Appl. 100, 234–245 (2018)
8. F. Carcillo, Y.-A. Le Borgne, O. Caelen, Y. Kessaci, F. Oblé, G. Bontempi, Combining unsupervised and supervised learning in credit card fraud detection. Inf. Sci. 557, 317–331 (2021)
9. N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique. J. Artif. Intell. Res. 16, 321–357 (2002)
10. A. Dal Pozzolo, O. Caelen, R.A. Johnson, G. Bontempi, Calibrating probability with undersampling for unbalanced classification. In IEEE Symposium Series on Computational Intelligence (SSCI), pp. 159–166 (2015)
11. U. Fiore, A. De Santis, F. Perla, P. Zanetti, F. Palmieri, Using generative adversarial networks for improving classification effectiveness in credit card fraud detection. Inf. Sci. 479, 448–455 (2019)
12. T. Saito, M. Rehmsmeier, The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets. PLoS ONE 10(3), e0118432 (2015)
13. V. Nair, G.E. Hinton, Rectified linear units improve restricted Boltzmann machines. In Proceedings of the 27th International Conference on Machine Learning (ICML), pp. 807–814 (2010)
14. D.P. Kingma, J. Ba, Adam: a method for stochastic optimization. arXiv:1412.6980 (2014)
15. T. Fawcett, An introduction to ROC analysis. Pattern Recognit. Lett. 27(8), 861–874 (2006)
16. F. Pedregosa et al., Scikit-learn: machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830 (2011)