# Credit Card Fraud Detection Using Hybrid Proximal Policy Optimization and Artificial Bee Colony Optimization with Mutual Learning

Yuanyuan Zhang
College of Economic and Management, North China Institute of Science and Technology وLangfang, Hebei 065201, China
E-mail: zhangyy4935@163.com

**Keywords:** credit card fraud detection, proximal policy optimization, imbalanced learning, artificial bee colony, artificial neural network

**Received:** Januar 19, 2025

*The surge in e-commerce has intensified credit card fraud, resulting in massive global losses and creating an urgent need for stronger detection systems. This research presents an advanced model for detecting credit card fraud to address challenges often overlooked, such as class imbalance and sensitivity to initial parameter settings. Our model leverages an artificial neural network (ANN) to extract feature vectors necessary for accurate fraud detection. We utilize proximal policy optimization (PPO) to address class imbalance during training of the ANN. PPO improves the treatment of minority classes by assigning higher rewards for correct predictions and more substantial penalties for errors. This approach leads to more balanced learning. Additionally, our model incorporates a mutual learning-based artificial bee colony (ML-ABC) algorithm for efficiently pre-training the parameters of the ANN. Experiments on the Université Libre de Bruxelles credit card dataset show that the proposed approach achieves 90.197% accuracy and an F-measure of 91.287%. It outperforms the best existing method by about 3%. These results highlight the robustness of the model and its potential for real-world e-commerce fraud detection.*

*Povzetek: Raziskava predstavlja napreden model za zaznavanje goljufij s kreditnimi karticami, ki izboljšuje natančnost in presega obstoječe metode.*

## 1 Introduction

Credit card fraud poses a significant threat to e-commerce, resulting in substantial financial losses for both businesses and consumers. Reports indicate that in the first quarter of 2018, fraud in e-commerce transactions increased at a faster rate than overall transaction volumes in 2016 [1]. The E-commerce Fraud Index reported a sharp rise in account takeover cases in online department stores. Rates increased from 0.06% in 2016 to 0.23% in 2017, contributing to more than 10% of total fraud losses [2]. Although credit card fraud represents only 0.1% of all card transactions, the large amounts involved make it highly damaging [3]. Therefore, an automated fraud detection system is essential. This paper presents a deep learning–based method for detecting credit card fraud. The method addresses two challenges: class imbalance and sensitivity to initial parameter settings.

Various approaches are used to detect credit card fraud, including statistical, machine learning (ML), and deep learning (DL) methods [4–6]. A key challenge in fraud detection is data imbalance, where legitimate transactions far outnumber fraudulent ones. This imbalance often biases model predictions and reduces the effectiveness of detection. To address this, researchers use both data-centric and algorithmic strategies [7]. Data-level techniques such as down-sampling and up-sampling aim to balance class distribution. Algorithm-level methods focus on improving the performance of minority classes. At the data level, skewed transaction distributions emphasize common patterns, sidelining rare but critical fraud cases. From an algorithmic perspective, the main challenge is to refine learning to capture rare but essential data. This refinement improves predictive accuracy and reliability. Deep reinforcement learning (DRL) has proven effective in managing class imbalances by filtering irrelevant data and emphasizing key features. Its strength lies in reward-based adaptation, enabling models to focus on underrepresented classes. By refining reward functions, DRL improves its sensitivity to critical yet overlooked cases. However, DRL also faces the bias-variance trade-off. To solve this problem, we use an advanced version of DRL, i.e., PPO. PPO is an advanced on-policy reinforcement learning algorithm. It uses a clipping mechanism to stabilize training and prevent excessive policy updates, ensuring efficiency in complex, continuous-variable environments.

DL models often use backpropagation to adjust weights and reduce errors. Despite its effectiveness, backpropagation is sensitive to initial weight settings and can become trapped in local minima, especially in classification tasks. To overcome these issues, researchers use meta-heuristic algorithms. Examples include the arithmetic optimization algorithm (AOA) [8], puma

optimizer (PO) [9], sparrow search algorithm (SSA) [10], chaotic sand cat swarm optimization (CSCSO) [11], whale optimization algorithm (WOA) [12], and political sand cat swarm optimization (PSCSO) [13]. These approaches enhance the search process, enabling models to avoid getting stuck in local minima. Among them, the ABC algorithm is particularly promising. The ABC algorithm is inspired by honey bee foraging. It balances exploration and exploitation in the search space, enabling the discovery of global optima in complex optimization tasks. It is effective in managing high-dimensional data and avoiding premature convergence, improving robustness and generalization. Furthermore, this study applies an advanced version, ML-ABC, which incorporates mutual learning to address weight initialization issues in gradient-based methods. By promoting information exchange, ML-ABC enhances adaptability and ensures more reliable optimization outcomes.

This study examines whether combining PPO reinforcement learning with ML-ABC optimization improves credit card fraud detection in imbalanced datasets. It focuses on three key elements:

- Research question: Can PPO and ML-ABC together outperform traditional ML and DL models in handling class imbalance for fraud detection?
- Hypotheses: 1) PPO increases sensitivity to minority (fraud) classes; 2) ML-ABC improves parameter initialization and lowers the risk of local minima.
- Goals: To design a model that (i) mitigates class imbalance, 1) stabilizes training through optimized initialization, and 2) improves robustness in fraud detection.

The key contributions of the dissertation are outlined below:

- PPO for imbalanced classification: The model uses PPO to address imbalanced credit card datasets. In such datasets, fraudulent transactions are much rarer than legitimate ones. PPO applies reward-based learning and clipping mechanisms. These methods direct attention to patterns in minority classes, increase sensitivity to fraud, and support stable training in dynamic environments.
- ML-ABC for initial weighting: The model introduces ML-ABC optimization for initializing network weights. This technique enhances exploration and exploitation in the solution space. It also prevents premature convergence and improves adaptability. ML-ABC provides optimized starting weights. This reduces vulnerability to local minima, leading to more reliable convergence and robust fraud detection.

The structure of this paper is organized as follows: Section 2 provides a literature review, while Section 3 describes our method for detecting credit card fraud. Section 4 presents the experimental outcomes, and Section 5 concludes the paper by summarizing the key findings.

# 2 Related works

This part provides an overview of credit card fraud recognition systems and methods developed in previous work. This area of investigation can be embodied in three groups: statistical, ML, and DL.

## 2.1 Statistical methods

Statistical methods are extensively applied to detect credit card fraud by analyzing the statistical patterns in transaction data [14]. These techniques focus on identifying anomalies by setting thresholds or following specific criteria. Common approaches include descriptive statistics, hypothesis testing, and time series analysis [15]. Descriptive statistics, such as mean, standard deviation, and percentiles, are used to detect irregular transactions [16]. Hypothesis testing distinguishes between legitimate and fraudulent transactions by using null and alternative hypotheses, employing statistical tests such as t-tests and chi-square tests [1]. Time series models, including ARIMA (AutoRegressive Integrated Moving Average) [17] and STL (Seasonal and Trend Decomposition using Loess) [18], are employed to uncover trends and patterns in transaction data that may signal fraud. These models help identify and predict deviations from normal behavior, thereby enhancing the proactive detection of fraudulent activities.

Although statistical methods are key in detecting credit card fraud, they face several limitations. These techniques depend heavily on precise threshold settings or predefined criteria. When thresholds are set too stringently, legitimate transactions may be incorrectly flagged as fraudulent, causing inconvenience and customer dissatisfaction. On the other hand, lenient thresholds may allow actual fraudulent activities to go unnoticed, increasing the risk of financial loss for institutions. Moreover, techniques such as descriptive statistics and hypothesis testing are based on assumptions about the distribution and behavior of transaction data, which may not always hold, especially given the dynamic and varied nature of financial transactions. Such assumptions can lead to false conclusions and the failure to detect fraud.

## 2.2 ML

ML algorithms have become indispensable in combating credit card fraud because they can learn from data, detect intricate patterns, and predict fraudulent activities.

In 2024, Khalid et al. [1] developed an ensemble ML system combining support vector machine (SVM), k-nearest neighbor (KNN), random forest (RF), bagging, and boosting. They used the synthetic minority over-sampling technique (SMOTE) and under-sampling to address the imbalance. Gajakosh et al. [2] proposed an unsupervised SVM model. The model is enhanced with competitive swarm optimization (CSO) to detect anomalies and reveal hidden fraud in online transactions. Suardiman et al. [19] developed a fraud detection model utilizing Gaussian Naïve Bayes, KNN, and RF algorithms to enhance the identification of fraudulent e-commerce transactions. Odeyale et al. [20] employed SMOTE and analysis of variance (ANOVA) F-statistics for data balancing and feature selection, thereby enhancing the performance of SVM on highly imbalanced credit card fraud datasets. Umalwara et al. [21] introduced a real-time

fraud detection system that utilizes adaptive ML (AML), enabling dynamic learning and rapid responses to unauthorized credit card transactions. Loukili et al. [22] implemented a detection system using categorical boosting (CatBoost), adaptive boosting (AdaBoost), and extreme gradient boosting (XGBoost). They compared the model's using precision, accuracy, and latency to evaluate performance in fraud detection.

In 2025, Salam et al. [23] implemented federated learning (FL) using TensorFlow Federated and PyTorch frameworks for credit card fraud detection (CCFD), integrating individual and hybrid resampling techniques to address class imbalance across distributed, privacy-preserving datasets. Kokate et al. [24] employed SVM models with various kernel functions, including the radial basis function (RBF), to classify credit card transactions as either fraudulent or legitimate based on historical transaction behavior patterns. Wang [25] proposed an ensemble ML model combined with the SMOTE using k-means clustering (SMOTE-KMEANS) to enhance fraud detection by addressing data imbalance in classification tasks. Baisholan et al. [26] developed an ensemble ML framework that integrates RF and XGBoost, applying probability averaging and threshold optimization to address fraud detection with interpretability using shapley additive explanations (SHAP).

## 2.3 DL

DL utilizes multi-layered neural networks that excel at capturing complex patterns and essential features within high-dimensional data, making it highly effective for fraud detection.

In 2023, Xie et al. [27] proposed a convolutional neural network (CNN) framework for detecting e-commerce fraud. It reduces parameters through local perception fields and weight sharing. Their model addresses overfitting, handles imbalanced datasets, and improves the prediction accuracy of fraudulent activities. Karthika et al. [28] introduced a one-dimensional dilated convolutional neural network (DCL) to detect credit card fraud. The model captures spatial-temporal features and mitigates data imbalance by utilizing under-sampling and over-sampling techniques, thereby enhancing detection accuracy. Balawi et al. [29] reviewed advancements in credit card fraud detection and classified prevention systems into categories. They evaluated ANN and CNN models, demonstrating that optimized deep neural networks enhance accuracy and reduce fraudulent activity in credit card datasets. Berhane et al. [30] developed a hybrid model that combines a CNN for feature extraction and an SVM for classification. They replaced the CNN output layer with an SVM, which improved fraud detection in online transactions. Fanai et al. [31] proposed a two-stage framework that utilizes a deep autoencoder for representation learning and supervised DL for fraud detection. This method enhanced classifier accuracy and robustness compared to training on raw or principal component analysis (PCA)-transformed data.

In 2024, Ida et al. [3] designed a long short-term memory (LSTM)–driven framework with an early stopping strategy to identify fraudulent credit card transactions. Their method leverages sequential transaction patterns, thereby reducing the risk of overfitting and enhancing model generalization. Suganthi and Jebathangam [32] employed a gated recurrent unit (GRU) model for fraud detection in mobile payments. The approach utilizes GRU to manage temporal data efficiently. It trains faster, detects fraud in real-time, and reduces computational costs compared to LSTM. Mienye and Swart [33] proposed a hybrid architecture that integrates a generative adversarial network (GAN) with a recurrent neural network (RNN). In this system, GAN generates synthetic samples. LSTM and GRU models then classify transactions as authentic or generated. Onyeoma et al. [34] analyzed deep neural networks (DNNs), including CNN, LSTM, RNN, multilayer perceptron (MLP), and deep belief network (DBN). Their work combines optimizers and explainability methods to increase fraud detection precision and improve interpretability. Senthilselvi et al. [35] compared several DL models, including RNN, CNN, ANN, and linear autoencoder, for credit card fraud analysis. By applying a soft voting strategy, they fused outputs to identify the most reliable detection mechanism. Ganj and Chaparala [36] developed a detection pipeline utilizing wavelet-based feature fusion, deep neuro-fuzzy networks, and the Zeiler and Fergus network (ZFNet) for classification. For hyperparameter fine-tuning, they applied the dwarf mongoose–shuffled shepherd political optimization (DMSSPO) algorithm. Veeru et al. [37] developed an e-commerce fraud detection approach. Their method uses PCA for preprocessing and autoencoders for feature extraction. A hybrid GRU-bidirectional LSTM (Bi-LSTM) classifier is applied, then optimized with coral reefs optimization (CRO) and secured through homomorphic encryption. Yu et al. [38] investigated transformer-based architectures for credit card fraud detection, benchmarking them against SVM, random forests, neural networks, logistic regression, XGBoost, and tabular networks (TabNet) methods.

In 2025, Dubey et al. [39] proposed a hybrid system that combines a transformer-based encoder, multi-teacher knowledge distillation, and symbolic belief–desire–intention (BDI) reasoning. This integration supports both deep feature learning and symbolic reasoning for fraud detection. Kandi et al. [40] implemented LSTM networks combined with XGBoost to enhance sequential and ensemble learning for credit card fraud detection. They addressed the imbalance in datasets using the SMOTE, improving detection outcomes. Yousefimehr et al. [41] proposed a fraud detection pipeline that merges one-class support vector machine (OCSVM), the SMOTE, and random undersampling. Their design was validated using LSTM and light gradient boosting machine (LightGBM) classifiers to improve prediction accuracy. Khine et al. [42] compared CNN for spatial representation and LSTM for temporal modeling. They applied a sampling strategy that better represents minority classes in imbalanced datasets, improving detection accuracy. Sultana et al. [43] presented a graph neural network (GNN) model enriched

with time-dependent attributes and adaptive updates. Their framework employs a lambda neural network to capture complex, layered relationships in credit card fraud patterns.

## 2.4 Limitations

Table 1 presents ML and DL algorithms for credit card fraud detection. It summarizes their contributions, datasets, performance, and limitations. The table illustrates a wide variety of strategies. These range from ensemble learning and optimization-based SVMs to advanced DL models, including CNNs, RNNs, GANs, and transformers. ML methods face challenges such as dependence on manual feature selection, difficulty in capturing sequential or contextual patterns, poor adaptability to dynamic data, and reduced ability to generalize under conditions of class imbalance.

DL has improved fraud detection by automatically extracting features, modeling sequential dependencies, and learning complex nonlinear patterns. However, DL methods still suffer from class imbalance and sensitivity to initial parameter settings. To address these issues, this article introduces a model that integrates PPO and ML-ABC. PPO provides adaptive handling of imbalanced data. It offers an alternative to oversampling methods, such as SMOTE, which can create redundant or unrealistic samples. ML-ABC ensures more reliable weight initialization and reduces the risk of local minima. The synergy produces balanced classification. It also supports stable training, leading to robust fraud detection performance. This synergy is considered a central innovation of this study.

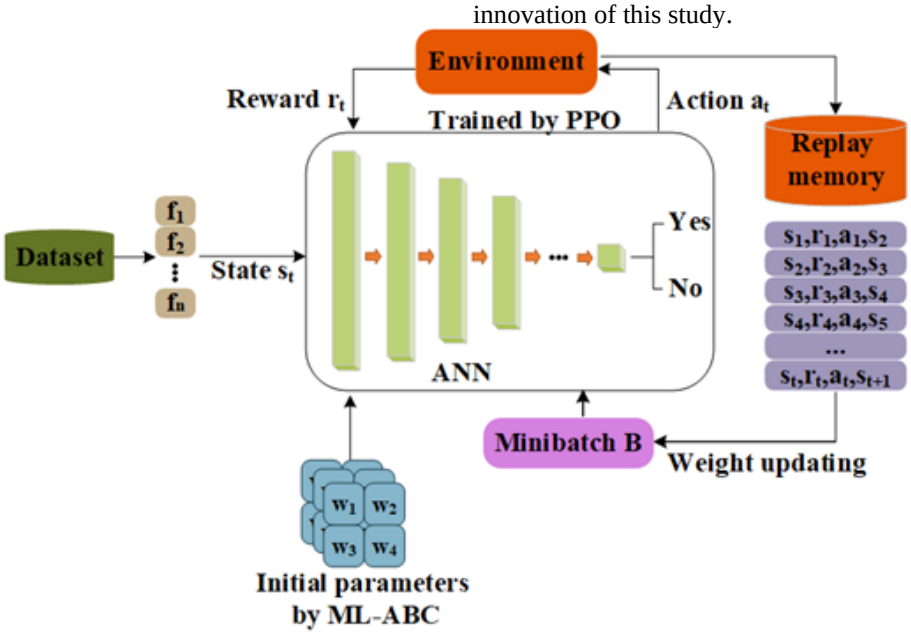


Figure 1: The proposed framework includes an ML-ABC for initial parameter setting and a PPO-based training for imbalanced classes.

# 3 Material and methods

This paper aims to enhance credit card fraud detection by addressing two persistent issues: inadequate initial weight setting and severe class imbalance. Figure 1 illustrates the architecture of the proposed model, which is based on an ANN. The input to the ANN is an $n$-dimensional vector. To initialize the ANN, the ML-ABC algorithm is used. ML-ABC generates optimized starting weights, reducing the risk of local minima and improving convergence stability. After setting the initial weight, the ANN is trained using the PPO algorithm. PPO adapts the learning process to focus on minority fraud cases. PPO uses a reward-based update strategy. This ensures stable training and increases sensitivity to rare but critical fraudulent transactions. The integration of ML-ABC and PPO provides a synergistic framework. Their combined use delivers reliable initialization, balanced classification, and improved robustness.

## 3.1 Initial weight using ML-ABC

In the conventional ABC procedure, the selection of food source locations begins randomly. Each chosen location is then altered to produce a candidate alternative. When this candidate demonstrates superior fitness, it replaces the current location; otherwise, the original choice remains unchanged. In multi-dimensional optimization, one dimension is selected at random, and its corresponding value is modified while keeping others fixed. At each step, the solution that performs better is carried forward. As shown in Equation 1, the generation of new candidates depends only on two parameters: $x_i^j$ and $x_k^j$. This limited reliance reduces the likelihood of consistently generating high-quality solutions. Some candidates improve the current solution, while others lead to a drop in fitness. The overall objective of the ABC process is to locate food sources that achieve higher fitness values.

$$v_i^j = x_i^j + \varphi_i^j (x_i^j - x_k^j) \quad (1)$$

To refine the search process, an advanced variant of the ABC algorithm is introduced. This version uses a mutual learning strategy where each candidate both shares and

Table 1: Summary of ML and DL algorithms for credit card fraud detection, highlighting contributions, datasets, performance, and key limitations.

| Authors | Method | Contribution | Dataset | Result | Limitation |
|---|---|---|---|---|---|
| Khalid et al. [1] | Ensemble (SVM, KNN, RF, Bagging, Boosting) | Resampling-based ensemble for imbalance | European credit card users (284,807 transactions) | Accuracy: 94.3% | Complex ensemble, low interpretability |
| Gajakosh et al. [2] | SVM (unsupervised) | CSO tuning | European credit card users (284,807 transactions) | Accuracy: 99.88% | Sensitive to kernel/CSO parameters |
| Suardiman et al. [19] | Gaussian NB, KNN, RF | Tri-model evaluation for e-commerce | European credit card users (284,807 transactions) | Accuracy: 99.5% | No imbalance-specific methods |
| Odeyale et al. [20] | SVM, SMOTE, and ANOVA | Balancing and feature selection | European credit card users (284,807 transactions) | Accuracy: 93.9% | Ignores nonlinear feature relations |
| Umalwara et al. [21] | AML | Real-time adaptive fraud detection | European credit card users (284,807 transactions) | Accuracy: 88.3% | Needs careful drift handling |
| Loukili et al. [22] | CatBoost, AdaBoost, and XGBoost | Comparative boosting with latency metrics | European credit card users (284,807 transactions) | Accuracy: 91.1% | Risk of overfitting without regularization |
| Salam et al. [23] | FL | Privacy-preserving distributed detection | European credit card users (284,807 transactions) | Accuracy: 99.98% | Client drift and high communication cost |
| Kokate et al. [24] | SVM (RBF kernel) | Kernel-based fraud classification | European credit card users (284,807 transactions) | Accuracy: 95% | Hyperparameter tuning critical |
| Wang [25] | Ensemble and SMOTE-KMEANS | Cluster-aware synthetic oversampling | European credit card users (284,807 transactions) | Area under the curve (AUC): 0.96 | Cluster assumptions may mislead |
| Baisholan et al. [26] | RF, XGBoost, and SHAP | Probability averaging with interpretability | European credit card users (284,807 transactions) | Accuracy: 99% | SHAP may miss feature interactions |
| Xie et al. [27] | CNN | Local perception and weight sharing for parameter reduction | Open-source e-commerce service data from 2018 (Kaggle) | Accuracy: 83.60% | Limited to spatial feature extraction |
| Karthika et al. [28] | Dilated CNN | Captures spatial-temporal features with dilation | UCSD–FICO data mining contest 2009 credit card dataset | Accuracy: 97.45% | Dilation may ignore fine-grained details |
| Balawi et al. [29] | ANN and CNN | Comparative study of ANN and CNN for fraud | Kaggle Credit-card Fraud dataset | Accuracy: 99.81% | No novel architecture proposed |

| | | | | | |
|---|---|---|---|---|---|
| | | | (284,807 transactions) | | |
| Berhane et al. [30] | CNN and SVM | Replaces CNN output with a SVM classifier | European credit card users (284,807 transactions) | Accuracy: 91.08% | SVM layer limits end-to-end learning |
| Fanai et al. [31] | Autoencoder and DNN | Two-stage feature learning and classification | European credit card users (284,807 transactions) | Accuracy: 89.22% | Training complexity increased |
| Ida et al. [3] | LSTM | Sequential learning with early stopping | European credit card users (284,807 transactions) | Accuracy: 92.06% | Dependent on time-sequenced data |
| Suganthi and Jebathangam [32] | GRU | Efficient real-time detection model | European credit card users (284,807 transactions) | Accuracy: 90.25% | GRU underperforms for long sequences |
| Mienye and Swart [33] | GAN and RNN | GAN-generated samples classified by RNNs | European credit card users (284,807 transactions) | F-meature: 90.6% | GAN instability risk |
| Onyeoma et al. [34] | DNN (CNN, LSTM, RNN, and MLP, DBN) | Combines optimizers with explainability methods | Kaggle comprehensive credit card transaction dataset (284,000) | F-meature: 99% | High model complexity and added overhead from explainability |
| Senthilselvi et al. [35] | RNN, CNN, and ANN | Model fusion via soft voting | European credit card users (284,807 transactions) | Accuracy: 99.4% | Voting may dilute strong learners |
| Ganj and Chaparala [36] | ZFNet and Neuro-fuzzy | Wavelet fusion and DMSSPO tuning | European credit card users (284,807 transactions) | Accuracy: 96.1% | High complexity in tuning |
| Veeru et al. [37] | PCA, autoencoder, GRU-BiLSTM, and CRO | Preprocessing, feature extraction, and a hybrid classifier with encryption | European credit card users (284,807 transactions) | Accuracy: 88.6% | Encryption and CRO increase inference latency |
| Yu et al. [38] | Transformer | Benchmarked with classic ML methods | European credit card users (284,804 transactions) | F-meature: 99.8% | Transformer training resource-intensive |
| Dubey et al. [39] | Transformer + BDI | Combines symbolic logic and DL | IEEE computational intelligence society fraud detection dataset | Accuracy: 90.9% | Symbolic integration adds design complexity |
| Kandi et al. [40] | LSTM and XGBoost | Uses SMOTE for class balance | Kaggle dataset | Accuracy: 97% | Relies heavily on oversampling |
| Yousefimehr et al. [41] | OCSVM, SMOTE, and LightGBM | Blends unsupervised and boosting | European credit card users (284,807 transactions) | F-meature: 87% | Limited interpretability from OCSVM |

| | | | | | |
|---|---|---|---|---|---|
| Khine et al. [42] | CNN and LSTM | Sampling-enhanced hybrid model | European credit card users (284,807 transactions) | F-meature: 91.6% | Sampling strategy sensitive to skew |
| Sultana et al. [43] | GNN | Adaptive GNN with time features | European credit card users (284,807 transactions) | Accuracy: 90.3% | GNNs are hard to scale on large data |

receives information from nearby solutions. Unlike the original ABC, which relies solely on random perturbations, the improved design directs updates through fitness-based comparisons. When a neighbor exhibits superior fitness, the current solution adapts and shifts toward it. If the current solution performs better, it is retained and slightly adjusted based on the influence of neighboring solutions. This dual approach improves exploitation by focusing the search near promising areas. At the same time, random variations ensure continued exploration of new regions. Guided updates reduce unnecessary randomness, accelerate convergence, and enhance the overall quality of generated solutions. As a result, the upgraded ABC framework becomes more stable and enables more reliable feature extraction. This leads to better performance in tasks involving initial weight generation.

The mutual learning mechanism used in this paper operates as described below:

$$v_i^j = \begin{cases} x_i^j + \varphi_i^j\left(x_k^j - x_i^j\right), & Fit_i < Fit_k \\ x_k^j + \varphi_i^j\left(x_i^j - x_k^j\right), & Fit_i \geq Fit_k \end{cases} \quad (2)$$

In this approach, the terms $Fit_i$ and $Fit_k$ represent the fitness values of the neighbor and the current candidate, respectively. The parameter $\varphi_i^j$ is a randomly chosen scalar drawn from the interval $(0, F)$, where $F$ is the mutual learning factor, constrained to be greater than zero. This mechanism promotes the adoption of traits from solutions with superior fitness scores. The mutual learning coefficient $F$ plays a key role in both refining solution quality and ensuring stability. A higher value of $F$ leads to smaller update fluctuations, effectively directing the search toward fitter neighbors. Nonetheless, excessively high values of $F$ can disturb the necessary exploration–exploitation trade-off, potentially reducing the overall effectiveness of the optimizer.

Figure 2 illustrates the workflow of the ABC algorithm enhanced with mutual learning. Initially, a swarm of bees is positioned randomly across the search domain. In the employed bee stage, each bee inspects a neighboring candidate and compares its fitness. If the neighbor has higher fitness, the bee moves toward that candidate. The step length and direction depend on the difference in fitness between the two solutions. If the neighbor is weaker, the bee maintains its current position but applies small random changes to support local search. Bidirectional learning makes the bee movements more guided. This improves the balance between exploration and exploitation, helping the algorithm converge faster. The remaining stages follow the structure of the classical ABC. In the onlooker stage, bees select food sources based on fitness probabilities. They then update the selected sources using the same set of update rules. In the scout stage, unproductive solutions are replaced with new ones.

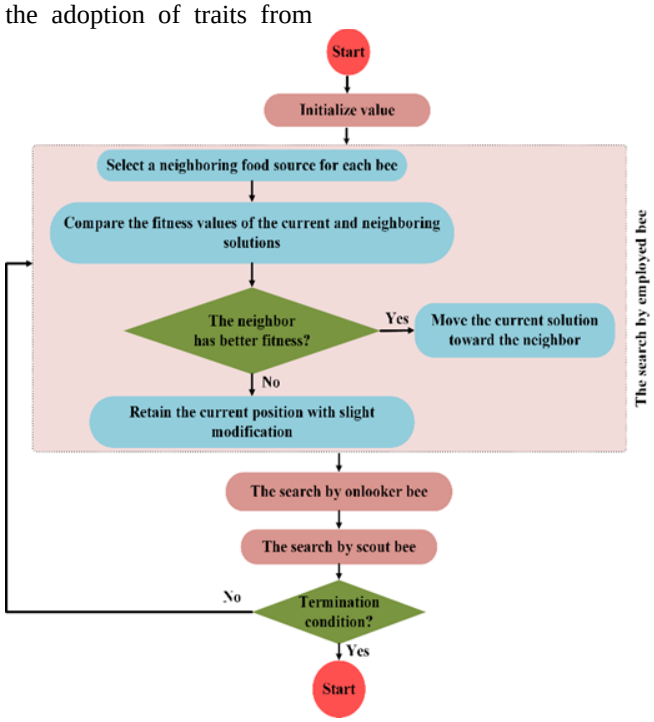


Figure 2: Overview of the ML-ABC algorithm phases.

Algorithm 1 describes the ML-ABC optimization process, which begins by initializing a population of num_bees candidate solutions within the range $[x_{min}, x_{max}]$. Each bee updates its position iteratively for max_cycles using three phases: employed, onlooker, and scout. During the employed bee phase, each bee evaluates its solution against a neighbor ($k \neq i$) and adjusts its position using a directional update. This update is controlled by the mutual learning coefficient F and a random factor $\varphi_i^j$. Onlooker bees select solutions probabilistically based on fitness and apply the same rule. This fitness-driven, F-scaled mutual learning promotes faster convergence while preserving global search capability.

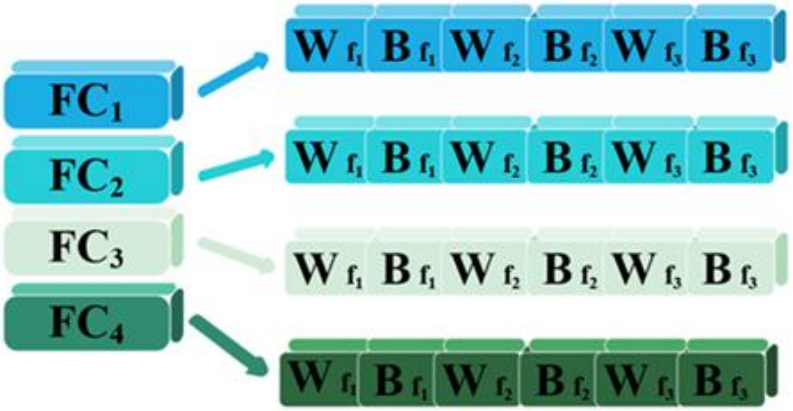


Figure 3: An example of the encoding technique in the ML-ABC algorithm.

### 3.1.1 Setting

Figure 3 shows an example of the encoding strategy used in this paper for an ANN. Here, the weight matrices within the array are organized in a row-wise format.

Fitness function calculates the quality of each candidate solution. In this paper, we propose an objective function based on similarity as:

$$Fitness = \frac{1}{1 + \sum_{i=0}^{N}(y_i - \tilde{y}_i)^2} \quad (3)$$

In this formula, $\boldsymbol{y_i}$ and $\tilde{\boldsymbol{y}}_{\boldsymbol{i}}$ represent the actual as well as forecasted labels for the i-th data instance, in that order, while N signifies the aggregate number of data instances.

## 3.2 Classification

In this paper, we use PPO to train the ANN. PPO improves policy performance in both discrete and continuous action spaces. The algorithm addresses the limits of earlier policy gradient methods, especially high sample demand and unstable learning. PPO applies small, incremental updates that avoid large deviations and keep the consistency of the policy within a safety margin. A clipped objective defines this margin and encourages small but beneficial changes, which help the agent accumulate reward.

This mechanism is formalized through the clipped surrogate objective of PPO, defined as follows:

$$L_{PPO}^{CLIP} = E_{s\sim\rho_{\pi_{old}},\ a\sim\pi_{old}}[\min(\frac{\pi(a|s)}{\pi_{old}(a|s)}A_{\pi_{old}}(s,a), clip(\frac{\pi(a|s)}{\pi_{old}(a|s)}, 1-\epsilon, 1+\epsilon)A_{\pi_{old}}(s,a))] \quad (4)$$

where $E$ denotes the expected value. $s$ represents the state of the environment, and $a$ represents the action taken by the agent. $\rho_{\pi_{old}}$ refers to the state distribution under the old policy $\pi_{old}$. The term $\epsilon$ is a small, positive constant that balances stability and exploration. The advantage function $A_{\pi_{old}}(s,a)$ indicates how much more reward the action $a$ in state $s$ yields compared to the average reward of all possible actions in that state under the old policy. Clipping the ratio limits drastic policy updates and stabilizes the learning process:

$$clip(x,a,b) = max(a, min(b,x)) \quad (5)$$

where $x$ is the value to be adjusted. $a$ is the minimum limit, and $b$ is the maximum limit of the range within which x is restricted. This design penalizes updates where the probability ratio diverges significantly from 1. Although PPO offers significant advantages, it also has a key limitation. It relies heavily on on-policy data, which leads to high sample complexity. This dependency necessitates frequent interaction between the agent and the environment, thereby increasing the computational cost of training.

To improve PPO performance and reduce sensitivity to poor initialization, this study integrates ML-ABC to pretrain the initial weights of the policy network (ANN) used in PPO. Rather than beginning PPO training with random parameters, ML-ABC first optimizes the network weights through mutual learning to offer a better starting point. This approach enhances sample efficiency and speeds up convergence. The integration enables PPO to begin training under more favorable conditions, thereby reducing the likelihood of poor policy updates during the early stages.

Algorithm 1: The ML-ABC algorithm

**Input:**
-$x_{min}$, $x_{max}$: lower and upper bounds for each solution dimension
-max_cycles: maximum number of optimization cycles
-num_bees: total number of bees (solutions) in the colony
-$F$: mutual learning coefficient
-$trial\_limit$: maximum allowed non-improving cycles before scout bee reinitializes the solution
**Output:**
- Best solution found
1: //Initialization population:
2: **for** $i = 1$ to num_bees **do**:
3: **for** $j = 1$ to $num\_dimensions$ **do**:
4: $x_i^j = x_{min}^j + rand(0,1) \times (x_{max}^j - x_{min}^j)$
5: **end for**
6: Evaluate the fitness of $x_i$
7: **end for**
8: $cycle = 1$
9: **while** $cycle <=$ max_cycles **do**:
10: //Employed Bee Phase:
11: **for** $i = 1$ to num_bees **do**:
12: Select $k \neq i$ randomly
13: **for** $j = 1$ to $num\_dimensions$ **do**:
14: $\varphi_i^j$ = rand (0, F)
15: **if** $fitness(x_i) < fitness(x_k)$ **then**:
16: $v_i^j = x_i^j + \varphi_i^j(x_k^j - x_i^j)$
17: **else**
18: $v_i^j = x_k^j + \varphi_i^j(x_i^j - x_k^j)$
19: **end if**
20: **end for**
21: Evaluate the fitness of $x_i$
22: **if** $fitness(x_i) < fitness(x_k)$ **then**:
23: $x_i = x_k$
24: **end if**
25: **end for**
26: // Onlooker Bee Phase
27: Calculate probabilities $p_i$ for each solution $x_i$ using
28: **for** $i = 1$ to num_bees **do**
29: Select $i$ based on $p_i$
30: Repeat steps 12–24 for selected $x_i$
31: **end for**
32: // Scout Bee Phase
33: **for** i = 1 to $num\_bees$ **do**:
34: **if** $x_i$ not improved for $trial\_limit$ **then:**
35: Repeat steps 12–24 for selected $x_i$
36: **end if**
37: **end for**
38: **end while**

### 3.2.1 Problem formulation

The state $\boldsymbol{s_t}$ in our model corresponds to the data sample drawn from the database at the $t$-th time step. The categorization that was done on the sample is reflected in categorization, a reward $r_t$ is assigned to every classification. The structure of this reward system is articulated as follows [44]:

the action $a_t$ and therefore symbolizes the judgment rendered by the network given its current knowledge. Furthermore, to guide this network towards more accurate

$$r_t(s_t, a_t, y_t) = \begin{cases} +1\,, if\ a_t = y_t\ and\ s_t \in D_O \\ -1\,, if\ a_t \neq y_t\ and\ s_t \in D_O \\ \lambda\,, if\ a_t = y_t\ and\ s_t \in D_N \\ -\lambda\,, if\ a_t \neq y_t\ and\ s_t \in D_N \end{cases} \quad (6)$$

In Equation 6, the reward function adjusts based on both class membership and prediction accuracy. A sample from the minority class $\boldsymbol{D_O}$ receives +1 for a correct prediction and −1 for an incorrect one. For the majority class samples $\boldsymbol{D_N}$, rewards are scaled by a factor $\lambda \in (0,1)$, reducing their impact on the learning process. Correct predictions receive $+\lambda$ and incorrect ones receive $-\lambda$. This asymmetric reward structure emphasizes fraudulent (minority) cases while maintaining a balanced learning approach. The use of $\lambda$ provides fine control over the sensitivity of the model to each class, improving performance on imbalanced datasets. This custom reward function is used in the PPO update process instead of the standard reward term when calculating the advantage. By penalizing or rewarding actions based on both class type and prediction accuracy, it allows PPO to prioritize learning from minority class samples without disrupting overall training stability.

Algorithm 2: Training the ANN model using PPO

**Input:**
$Data = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$: Training data
$E$: maximum number of iterations
$NM$: Number of minibatches
$\alpha$: learning rate
**Output:**
- Trained ANN model (policy network $\pi_\theta$)
1: Initialize $\pi_\theta$ by setting parameters $\theta$ using ML-ABC
2: Initialize a memory buffer $B$ for storing state transitions
3: **for** episode = $1:E$ **do**:
4: Shuffle the dataset $Data$
5: **for** $t = 1$ to $M$ **do**:
6: Observe the current state $s_t$
7: Select action $a_t \in \pi_\theta(a_t|s_t)$
8: Compute reward $r_t$ based on Equation 6
9: Store transition $(s_t, a_t, r_t, s_{t+1})$ in trajectory buffer $B$
10: **end for**
11: Estimate advantage values $A(s_t, a_t)$ using a value function $\phi$
12: **for** $k = 1:NM$ **do**:
13: Sample randomly a mini-batch $(s_k, a_k, r_k, s_{k+1})$ from B
14: Optimize the policy π to maximize:
15: $\theta \leftarrow \theta_{old} + \alpha \times \nabla_\theta L_{PPO}^{CLIP}(\pi)$
16: **end for**
17: **end for**

Algorithm 2 outlines the training procedure of the ANN model using PPO. Training starts by initializing model parameters using ML-ABC, which improves the starting conditions. A memory buffer collects transitions during each episode. At the start of each episode, the data is shuffled. The agent then interacts with the environment by observing states, choosing actions, and collecting rewards. These transitions are stored and used later to calculate advantage estimates. PPO loss is optimized using randomly sampled mini-batches from the buffer. This cycle continues over multiple iterations to gradually improve the performance of the policy network.

## 3.3 Time complexity analysis

The computational cost of the model comes from two main sources: (1) ML-ABC, which initializes ANN weights, and (2) PPO, which trains the ANN.

- **ML-ABC complexity:**

Let $D$ represent the number of dimensions in each solution vector, which corresponds to the total number of trainable weights in the ANN. Let $N$ denote the number of bees and $T$ the *maximum* number of optimization cycles. The core of the ML-ABC involves three phases: employed, onlooker, and scout. In each phase, the algorithm evaluates the fitness of solutions and updates the positions of some or all bees.

In each cycle, every bee updates all $D$ dimensions of its solution vector. The computational cost of each cycle is $O(N \times D)$, considering the operations in both the employed *and* onlooker bee phases. If none of the solutions improve, all bees may enter the scout phase, which introduces an additional $O(N \times D)$ operations. Therefore, the overall time complexity of ML-ABC is $O(T \times N \times D)$.

- **PPO complexity:**

Assume that PPO trains for $E$ episodes, on a dataset of size $M$, using $NM$ mini-batches per episode. Let $P$ represent the time *needed* for a single forward and backward pass through the ANN. This value depends on the number of parameters the network has. Each PPO update involves calculating policy ratios, applying the clipped objective, computing gradients, and running backpropagation. For each episode, PPO processes approximately NM batches. Hence, the PPO training complexity is: $O(E \times NM \times P)$

- **Total complexity:**

Combining both components, the overall time complexity becomes: $O(T \times N \times D) + O(E \times NM \times P)$. This indicates that the total computational cost depends mostly on the number of ANN parameters ($D$) and increases linearly with both the ML-ABC cycles ($N$) and PPO *training* episodes ($E$). Because ML-ABC generates optimized initial weights, PPO usually converges more quickly, which lowers its practical runtime. This synergy leads to efficient and stable training, particularly in scenarios involving imbalanced learning.

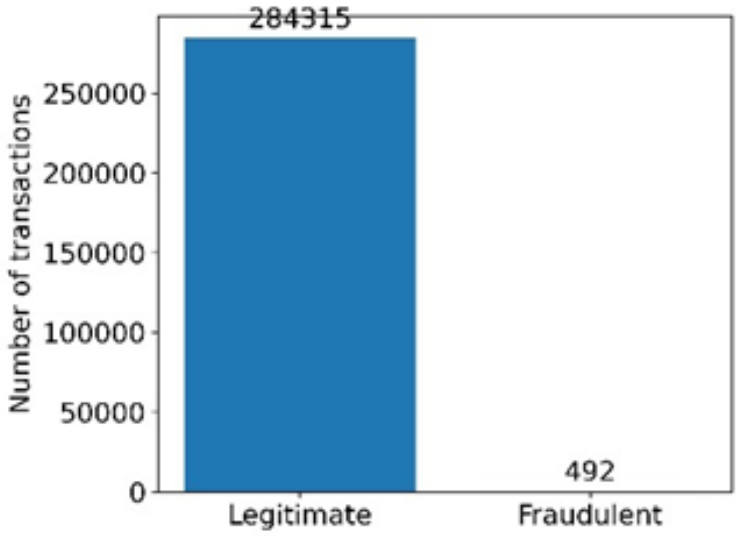


Figure 4: Distribution of legitimate vs. fraudulent transactions.

# 4 Experiment results

## 4.1 Database

The database utilized in this investigation is publicly available and was sourced from the Université Libre de Bruxelles. The dataset contains credit card transactions from European cardholders collected in September 2013. This database comprises 284,807 transactions by European cardholders over two days. Figure 4 shows a histogram of fraudulent and legitimate transactions, highlighting the class imbalance. Out of 284,807 transactions, 492 are labeled as fraudulent, accounting for only 0.172% of the entire dataset. This severe class imbalance makes the dataset highly representative of real-world fraud detection scenarios.

The dataset consists entirely of numerical features that have been transformed using PCA. V1–V28 are PCA-derived components. Time (seconds since the first transaction) *and* Amount (transaction value) are not transformed. Time and Amount are important for cost-sensitive fraud detection. The Class feature is the target variable, where 1 indicates fraud and 0 indicates legitimate transactions. For privacy reasons, raw attributes are not disclosed.

To ensure robust model training, we conducted exploratory data analysis (EDA) and preprocessing:

- Normalization: The Time and Amount features were normalized using z-score standardization.
- Stratified train-test split: An 80/20 split was applied, preserving class distribution to reduce variance across folds.
- Class imbalance check: We confirmed a severe imbalance (fraudulent class = 0.172%), which supports the use of imbalance-aware methods, such as PPO, in this study.

## 4.2 Metrics

To fairly evaluate the model under class imbalance, three metrics are used: accuracy, F-measure, and G-means. Accuracy gives a general performance overview but can be misleading with imbalanced data. F-measure balances precision and recall, highlighting the effectiveness of the model on the minority class, which is critical in fraud detection. G-means is the geometric mean of sensitivity and specificity. It measures how well the model performs on both classes and ensures that gains for the minority class do not reduce the accuracy on the majority class. Together, these metrics provide a comprehensive view of model performance, ensuring reliability and robustness for real-world fraud detection applications.

These metrics are defined as follows:

$$\text{Accuracy} = \frac{TP + TN}{Total\ number\ of\ samples} \tag{7}$$

$$F - measure\ = 2 \times \frac{Precision + Recall}{Precision + Recall} \tag{8}$$

$$\text{G} - \text{means} = \sqrt{\text{Recall} \times\ Specificity} \tag{9}$$

with

$$\text{Precision} = \frac{TP}{TP + FP} \tag{10}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{11}$$

$$Specificity = \frac{TN}{TN + FP} \tag{12}$$

In this context, $TP$ refers to the number of actual positive cases that are correctly identified by the system. $TN$ is defined as the actual negative conditions that are correctly predicted by the system. $FP$ refers to those instances where actual negative examples were falsely classified as positive, whereas $FN$ refers to actual positive examples that were misclassified as negative by the system.

## 4.3 Model performance

The experiments were conducted on a 64-bit Windows 11 system with 64 gigabytes (GB) of double data rate 4 (DDR4) random access memory (RAM) and an NVIDIA RTX 3090 graphics processing unit (GPU) (24 GB of video RAM (VRAM)). The GPU significantly accelerated training and evaluation of the deep neural networks. The software environment was based on Python 3.10, utilizing TensorFlow 2.13 and PyTorch 1.13.1, both compiled with Compute Unified Device Architecture (CUDA) version 11.7 to enable GPU support. PPO was implemented using PyTorch, while the ML-ABC algorithm was coded in Python using NumPy and SciPy. Efficient training on large, high-dimensional datasets from Université Libre de Bruxelles was achieved through a combination of fast memory access, GPU parallelism, and optimized ML libraries. This setup enabled thousands of training iterations with fast convergence and minimal delays during parameter tuning and evaluation.

Table 2: Hyperparameter configuration of the proposed model, determined via stratified 5-fold cross-validation.

| Hyperparameter | Value |
|---|---|
| Batch size | 73 |
| Epoch | 214 |
| Learning rate | 0.001 |
| Activation function | ReLU |
| Number of layers in ANN | 8 |
| Dropout rate | 0.56 |
| Clip range ($\in$) | 0.23 |
| Policy update steps | 5 |
| $\lambda$ | 0.4 |
| $F$ | 0.3 |

To set the hyperparameters of the proposed model, we use stratified 5-fold cross-validation. This method maintains the original class distribution in each fold, which is particularly important for imbalanced datasets such as those used in fraud detection. The dataset is divided into five equal parts, and each part maintains the same proportion of minority and majority classes. In each iteration, four folds are used for training and one for validation, rotating across all folds. Stratified sampling ensures a more consistent evaluation across folds and reduces variance caused by random splits. We chose five folds as a balance between statistical reliability and computational efficiency. We did not use repeated or nested cross-validation because they are computationally more expensive and provided little performance gain in our initial experiments. Table 2 shows the hyperparameter settings for the proposed model.

We compared the proposed model with four ML models and eight DL models. The ML models include SMOTE-Bagging-SVM [1], SVM-CSO [2], FL-Resampling [23], and XGBoost-SHAP [26]. The DL models include DCL [28], CNN-SVM [30], Bi-LSTM-CRO [37], TabNet [38], Transformer [39], LSTM-LightGBM [41], CNN-LSTM [42], and GNN [43]. The hyperparameters of the advanced baseline models were adopted directly from their original papers to ensure a fair comparison. We used a classification threshold of 0.5 for all binary decisions. This value is commonly used in fraud detection and ensures a fair and interpretable evaluation. Preliminary tests with alternative thresholds did not show significant improvements, confirming 0.5 as a balanced and widely accepted choice. The results of these tests are detailed in Table 3.

Table 3: Performance comparison of the proposed model against ML and DL baselines.

| Model | Accuracy | F-measure | G-means |
|---|---|---|---|
| SMOTE-Bagging-SVM [1] | 76.301 ± 0.019 | 77.878± 0.072 | 75.643± 0.073 |
| SVM-CSO [2] | 77.452 ± 0.025 | 78.749± 0.043 | 77.023± 0.010 |
| FL-Resampling [23] | 78.508 ± 0.079 | 78.902± 0.051 | 77.391± 0.028 |
| XGBoost-SHAP [26] | 79.117 ± 0.034 | 79.328± 0.059 | 78.086± 0.004 |
| DCL [28] | 81.362 ± 0.074 | 82.939± 0.100 | 80.672± 0.009 |
| CNN-SVM [30] | 82.389 ± 0.032 | 82.876± 0.046 | 81.612± 0.073 |
| Bi-LSTM-CRO [37] | 83.204 ± 0.002 | 83.493± 0.009 | 82.217± 0.079 |
| TabNet [38] | 83.824 ± 0.032 | 84.222± 0.043 | 83.181± 0.067 |
| Transformer [39] | 84.757 ± 0.089 | 84.835± 0.017 | 83.580± 0.044 |
| LSTM-LightGBM [41] | 85.324 ± 0.090 | 85.717± 0.067 | 84.461± 0.049 |
| CNN-LSTM [42] | 86.094 ± 0.008 | 86.701± 0.012 | 85.457± 0.088 |
| GNN [43] | 87.584 ± 0.055 | 87.827 ± 0.086 | 86.590 ± 0.079 |
| Proposed w/o ML-ABC | 81.871 ± 0.097 | 79.448± 0.032 | 81.095± 0.019 |
| Proposed w/o PPO | 86.981 ± 0.059 | 85.311± 0.018 | 86.943± 0.061 |
| Proposed | 90.197 ± 0.014 | 91.287± 0.070 | 89.456± 0.069 |

When comparing state-of-the-art models, the GNN model achieved the best performance among DL-based models, outperforming earlier approaches, such as DCL, by 6.222% in accuracy, 4.888% in F-measure, and 5.918%

in G-means. Similarly, GNN surpassed CNN-SVM by 5.195%, 4.951%, and 4.978% across the same metrics. However, GNN models are often challenging to scale and require substantial computational resources, thereby limiting their practicality. The Transformer also outperformed Bi-LSTM-CRO by 1.553%, 1.342%, and 1.363%, reflecting the benefits of attention mechanisms. Yet, transformer-based approaches generally demand extensive memory and training time, which reduces efficiency in real-world tasks. Among ML models, XGBoost-SHAP offered better interpretability but fell behind deep models, with an accuracy 8.467% lower compared to Transformer. Its limitation lies in weaker adaptability under severe imbalance despite strong interpretability. Although SVM-CSO used optimization, it still performed worse than CNN-SVM, with 10.132% lower accuracy and 9.078% lower G-means. This shows that parameter tuning alone cannot overcome the structural weaknesses of classical ML models. This gap highlights the advantage of hybrid DL methods over optimized ML models.

The proposed model consistently outperformed all ML and DL baselines. The proposed model surpassed the best-performing DL model, GNN, by 2.613% in accuracy, 3.460% in F-measure, and 2.866% in G-means. Relative to LSTM-LightGBM, it improved by 4.873%, 5.570%, and 4.995%, respectively. Compared to Transformer, the improvements were 5.440%, 6.452%, and 5.876%. Even over CNN-LSTM, it achieved a 4.758% gain in accuracy. When compared to the best-performing ML model, XGBoost-SHAP, the proposed model delivered even larger gains, 11.080% in accuracy, 11.959% in F-measure, and 11.370% in G-means. These consistent gains highlight the advantage of integrating ML-ABC for optimal initialization with PPO for stable training. Together, they improve generalization, handle class imbalance more effectively, and enable deeper feature abstraction than ensemble methods or single deep networks.

In ablation studies, compared to the version without PPO, we observed gains of 3.216% in accuracy, 5.976% in F-measure, and 2.513% in G-means. Against the version without ML-ABC, the proposed model achieved improvements of 8.326% in accuracy, 11.839% in F-measure, and 8.361% in G-means. A more detailed comparison shows that the accuracy of the proposed model improved by 2.084%, 3.216%, and 3.254% over PPO-only versions of CNN-SVM, Bi-LSTM, and TabNet, respectively. F-measure showed gains of 5.452%, 4.794%, and 5.065%, while G-means increased by 4.331%, 4.726%, and 6.275%. These findings indicate that PPO and ML-ABC alone are insufficient. Only their combination in the full model leads to the performance levels required for practical fraud detection.

To determine the statistical significance of the superiority of our proposed model over existing baselines, we employed paired t-tests across all three-evaluation metrics (accuracy, F-measure, and G-means). For both the best-performing ML model (XGBoost-SHAP) and the DL model (GNN), the results showed statistically significant differences, with all p-values below 0.001. Additionally, we computed 95% confidence intervals and effect sizes using Cohen's d. The effect sizes ranged from 1.20 to 2.85, indicating substantial differences. The 95% confidence intervals for accuracy gains spanned [2.14%, 4.92%], for F-measure [3.25%, 6.84%], and for G-means [2.88%, 5.67%]. For all other comparisons, p-values remained below 0.001, effect sizes ranged from 1.20 to 2.85, and the confidence intervals consistently demonstrated clear margins of improvement, supporting the robustness and reliability of the performance gains of the proposed model.

Table 4 presents a comparison of the computational efficiency of the proposed model with that of ML and DL models. The comparison includes runtime, GPU memory usage, and inference time per sample (ITPS). The proposed model achieved a runtime of 1984 seconds, which is 24.4% faster than LSTM-LightGBM, 15.9% faster than CNN-SVM, and 15.4% faster than TabNet. It also reduced GPU memory usage by 22.9% compared to LSTM-LightGBM and by 17.4% compared to Bi-LSTM-CRO. In terms of ITPS, the proposed model achieved 18.113 ms, which is 34.7% faster than Transformer and 28.8% faster than GNN. The proposed model is more efficient than DL baselines, but it still runs slower than simpler ML methods like XGBoost-SHAP. This illustrates a trade-off between accuracy and computational efficiency. The model is suitable for large-scale and real-time use, but may still require further optimization on limited hardware.

Table 4: Comparison of computational efficiency across models, including runtime, GPU memory usage, and ITPS.

| Model | Runtime (s) | GPU (GB) | ITPS (ms) |
|---|---|---|---|
| SMOTE-Bagging-SVM [1] | 1921 | 20.468 | 17.793 |
| SVM-CSO [2] | 1749 | 20.140 | 17.822 |
| FL-Resampling [23] | 1895 | 20.321 | 16.728 |
| XGBoost-SHAP [26] | 1777 | 20.878 | 15.837 |
| DCL [28] | 2348 | 23.622 | 26.200 |
| CNN-SVM [30] | 2817 | 23.110 | 22.182 |
| Bi-LSTM-CRO [37] | 2506 | 27.652 | 24.448 |
| TabNet [38] | 2587 | 23.431 | 27.330 |
| Transformer [39] | 2644 | 22.979 | 25.063 |
| LSTM-LightGBM [41] | 2744 | 28.246 | 27.739 |
| CNN-LSTM [42] | 2219 | 25.083 | 24.348 |
| GNN [43] | 2410 | 22.650 | 27.417 |
| Proposed | 1984 | 21.742 | 18.113 |

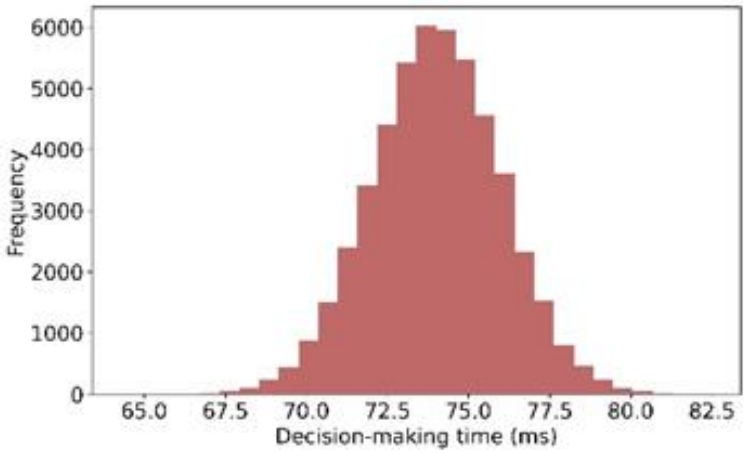


Figure 5: Distribution of decision-making time for the proposed model.

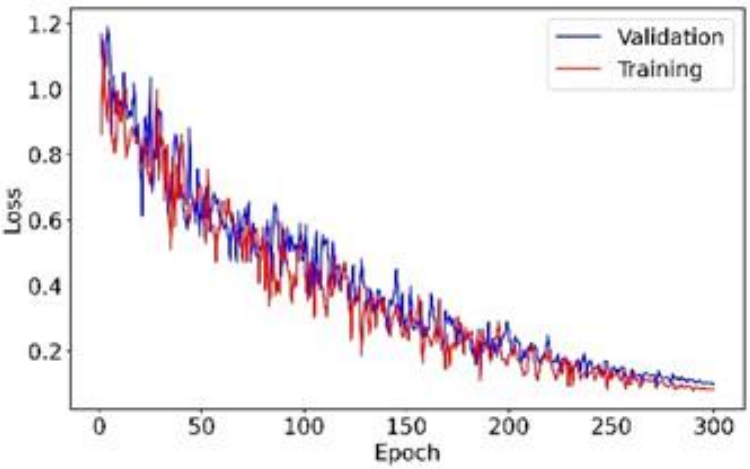


Figure 6: Training and validation loss curve in the proposed model.

Figure 5 illustrates the decision-making time of the proposed model, which supports its suitability for real-time applications. Most decision times are tightly grouped near 74 ms, with more than 95% falling within the 70- to 78-ms range. The small variation and fast decision speed demonstrate that the model provides quick and stable results, even under varying conditions, which is crucial for real-time fraud detection.

Figure 6 shows that both training and validation losses decrease steadily and remain close to each other throughout the epochs, without diverging. This means the model learns well, does not overfit, and converges properly.

Table 5: Performance comparison of the proposed model against ML and DL baselines on the IEEE-CIS dataset.

| Model | Accuracy | F-measure | G-means |
|---|---|---|---|
| SMOTE-Bagging-SVM [1] | 77.511 ± 0.054 | 78.202 ± 0.013 | 76.929 ± 0.025 |
| SVM-CSO [2] | 79.505 ± 0.023 | 80.220 ± 0.058 | 78.921 ± 0.065 |
| FL-Resampling [23] | 79.615 ± 0.055 | 79.891 ± 0.023 | 78.990 ± 0.035 |
| XGBoost-SHAP [26] | 81.812 ± 0.037 | 80.018 ± 0.079 | 79.731 ± 0.084 |
| DCL [28] | 82.588 ± 0.027 | 83.716 ± 0.083 | 81.427 ± 0.014 |
| CNN-SVM [30] | 83.288 ± 0.025 | 83.486 ± 0.011 | 82.207 ± 0.026 |
| Bi-LSTM-CRO [37] | 83.821 ± 0.017 | 84.438 ± 0.027 | 83.122 ± 0.077 |
| TabNet [38] | 84.355 ± 0.044 | 85.458 ± 0.001 | 84.154 ± 0.002 |
| Transformer [39] | 85.084 ± 0.012 | 85.520 ± 0.078 | 84.381 ± 0.051 |
| LSTM-LightGBM [41] | 86.601 ± 0.013 | 86.866 ± 0.001 | 85.824 ± 0.016 |
| CNN-LSTM [42] | 86.841 ± 0.015 | 87.096 ± 0.040 | 86.253 ± 0.009 |
| GNN [43] | 88.296 ± 0.094 | 88.939 ± 0.023 | 87.607 ± 0.083 |
| Proposed | 93.803 ± 0.029 | 92.890 ± 0.087 | 91.572 ± 0.007 |

### 4.3.1 Analysis of generalizability

To assess generalizability, robustness, and real-world applicability, we utilized the Institute of Electrical and Electronics Engineers Computational Intelligence Society (IEEE-CIS) dataset [35]. It is a large-scale, industry-grade benchmark featuring high-dimensional features, severe class imbalance, and noisy transaction records that accurately mirror real-world deployment scenarios. The dataset includes 569,875 legitimate transactions and 20,665 fraudulent ones. Fraud accounts for 3.626% of all transactions.

The results are shown in Table 5. The proposed model achieved 93.803% accuracy, 92.890% F-measure, and 91.572% G-means. Compared to the best ML baseline, XGBoost-SHAP, it improved by 12.184% in accuracy, 12.872% in F-measure, and 11.841% in G-means. Against the strongest DL model, GNN, it achieved gains of 5.507%, 3.951%, and 3.965%, respectively. These improvements over both ML and DL baselines show that the proposed hybrid design generalizes well. It maintains strong performance even when faced with extreme imbalance and noisy real-world transaction records.

### 4.3.2 Analysis of PPO

Class imbalance in fraud detection is often addressed through oversampling (e.g., SMOTE), undersampling, and cost-sensitive learning. These methods either rebalance the data or raise penalties for errors on the minority class. Despite their utility, oversampling risks overfitting, undersampling risks information loss, and cost-sensitive learning need careful weight tuning.

For ANNs, a common approach is to modify the loss function to handle class imbalance. In this study, we explored weighted cross-entropy (WCE), balanced cross-entropy (BCE), dice loss (DL), Tversky loss (TL), and combo loss (CL). WCE and BCE assign equal weight to positive and negative cases, while DL and TL balance false positives and false negatives. The CL mechanism lowers the weight of easy cases and emphasizes harder ones, making it effective for skewed distributions.

As shown in Table 6, CL reduced precision discrepancy by 31% and increased the F-measure by 42% compared to TL. However, PPO still surpassed CL by 71%. This large margin shows that advanced loss functions only partially address imbalance. In contrast, PPO offers a more dynamic and adaptive solution. By continuously shaping rewards, PPO prioritizes minority cases without reducing stability, outperforming both data-level and loss-level strategies.

Table 6: Performance comparison of PPO against established loss functions for imbalanced classification.

| Technique | Accuracy | F-measure | G-means |
|---|---|---|---|
| WCE | 75.445 ± 0.100 | 74.365 ± 0.087 | 76.006 ± 0.087 |
| BCE | 80.191 ± 0.078 | 77.030 ± 0.006 | 81.650 ± 0.099 |
| DL | 81.971 ± 0.020 | 80.984 ± 0.085 | 82.559 ± 0.022 |
| TL | 83.095 ± 0.001 | 81.779 ± 0.000 | 84.342 ± 0.066 |
| CL | 86.032 ± 0.092 | 84.291 ± 0.094 | 86.850 ± 0.013 |
| PPO | 90.197 ± 0.014 | 91.287± 0.070 | 89.456± 0.069 |

### 4.3.3 Analysis of ML-ABC

In this section, ML-ABC is compared with several well-known metaheuristic algorithms, including human mental search (HMS), firefly algorithm (FA), bat algorithm (BA), differential evolution (DE), grey wolf optimizer (GWO), cuckoo optimization algorithm (COA), AOA, PO, SSA, and ABC. For all metaheuristic algorithms, we set the population size to 50 and the maximum number of iterations to 512; other parameters are listed in Table 8 of Appendix A. These settings follow established literature. We did not use stratified cross-validation because repeated metaheuristic runs would greatly increase computational cost and make tuning impractical.

Table 7 shows that ML-ABC significantly outperforms the alternatives across all metrics. Compared to its base version, ABC, ML-ABC yields 5.876% higher accuracy, 7.564% higher F-measure, and 6.998% higher G-mean. Against PO, improvements are 8.015%, 9.000%, and 9.452%, respectively. Over AOA, ML-ABC offers a gain of 12.308% in accuracy and 11.919% in F-measure. The consistent superiority, especially over PO, SSA, and AOA, shows that the mental learning strategy improves both the direction of search and the diversity of candidate solutions. These improvements lead to better parameter initialization and more stable convergence during training. As a result, ML-ABC is a reliable option for complex classification tasks such as fraud detection. In such tasks, poor parameter initialization can negatively affect convergence and reduce final accuracy.

Table 7: Performance comparison of ML-ABC against established metaheuristic algorithms for initial parameter setting.

| Model | Accuracy | F-measure | G-means |
|---|---|---|---|
| HMS | 86.393 ± 0.063 | 85.883 ± 0.070 | 83.695 ± 0.014 |
| FA | 85.275 ± 0.005 | 83.690 ± 0.091 | 81.460 ± 0.070 |
| BA | 83.152 ± 0.017 | 82.448 ± 0.067 | 79.198 ± 0.098 |
| DE | 82.148 ± 0.018 | 81.517 ± 0.039 | 77.226 ± 0.071 |

| | | | |
|---|---|---|---|
| GWO | 80.791 ± 0.059 | 78.655 ± 0.031 | 75.358 ± 0.060 |
| COA | 75.681 ± 0.036 | 74.547 ± 0.079 | 70.254 ± 0.077 |
| AOA | 80.318 ± 0.015 | 79.224 ± 0.020 | 78.936 ± 0.038 |
| PO | 82.182 ± 0.099 | 81.287 ± 0.079 | 80.004 ± 0.049 |
| SSA | 83.765 ± 0.036 | 82.918 ± 0.054 | 81.627 ± 0.062 |
| ABC | 84.320 ± 0.043 | 83.723 ± 0.050 | 82.458 ± 0.049 |
| ML-ABC | 90.197 ± 0.014 | 91.287± 0.070 | 89.456± 0.069 |

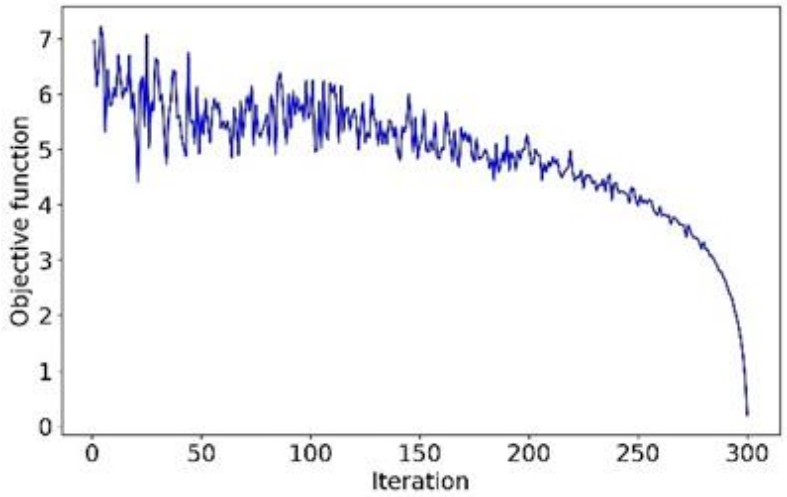


Figure 7: Convergence curve of ML-ABC showing the reduction in the objective function over 300 iterations.

Figure 7 presents the trend of the objective function across 300 iterations for the ML-ABC algorithm. The curve demonstrates a clear downward trajectory, with a significant reduction after iteration 250, indicating strong convergence behavior. The gradual decline followed by a sharp drop indicates that ML-ABC finds better solutions over time and converges efficiently.

### 4.3.4 Analysis of hyperparameter sensitivity

We used stratified 5-fold cross-validation to tune the model hyperparameters. Figure 8 shows a sensitivity analysis of important hyperparameters, including batch size, epoch, λ, and the number of layers in the ANN. The cross-validation process consistently identified the best-performing settings: batch size = 73, epoch = 214, λ = 0.4, and 8 layers. Higher or lower values than the identified optima led to a noticeable drop in performance. Stratification preserved class ratios in every fold and produced a stable, unbiased evaluation. This systematic approach confirmed model robustness and guided the choice of optimal parameters.

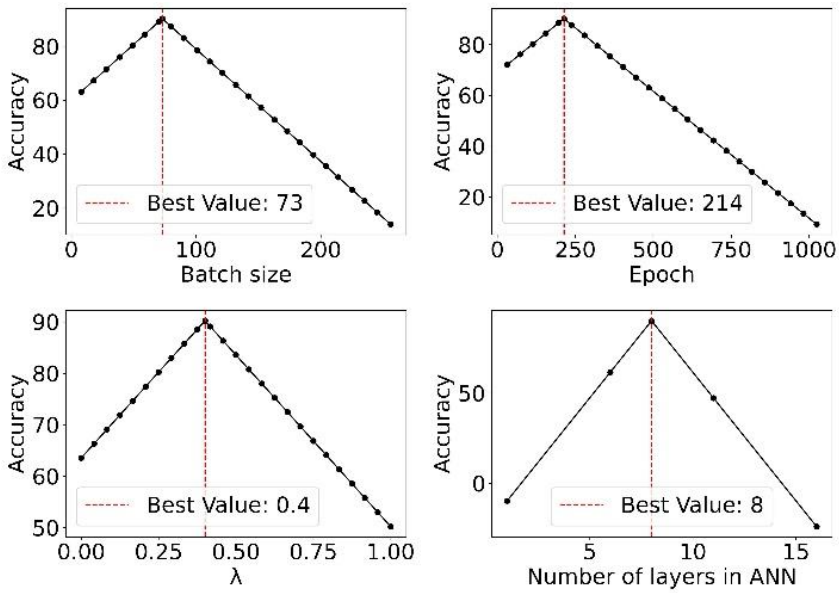


Figure 8: Sensitivity analysis of key hyperparameters (batch size, epoch, λ, and ANN layers) using stratified 5-fold cross-validation.

### 4.3.5 Discussion

This article introduces a hybrid fraud detection model that combines an ANN with ML-ABC for weight initialization and PPO for handling class imbalance. The proposed approach surpassed state-of-the-art models, achieving 90.197% accuracy, 91.287% F-measure, and 89.456% G-means on the Université Libre de Bruxelles dataset. These findings directly answer the research question, confirming that PPO and ML-ABC together outperform traditional ML and DL models. The results also confirm the hypotheses: PPO increases sensitivity to minority fraud cases, and ML-ABC improves parameter initialization while lowering the risk of local minima.

In our model, ML-ABC was used to improve ANN initialization. Traditional random starts often caused poor convergence, but mutual learning guided the process toward stable weights and reliable fraud detection. PPO addressed severe class imbalance by dynamically adjusting rewards to prioritize minority cases. Its clipped objective stabilized training, ensuring balanced learning and consistent performance across skewed datasets.

The proposed framework extends beyond credit card fraud to domains with imbalance and initialization challenges. In healthcare, it helps with early disease diagnosis, where ML-ABC stabilizes training and PPO balances rare conditions. In cybersecurity, it detects anomalies in network traffic to facilitate a quick response to intrusions, while in insurance, it enhances the detection of fraudulent claims. Its efficiency supports real-time decision-making. This makes it useful in financial risk management and industrial quality control. Its adaptability highlights its strength for high-accuracy tasks under imbalance with real-time requirements.

The limitations of the proposed model are as follows:

- One limitation of the proposed model is that it does not explicitly address feature selection. ANNs can capture complex nonlinear relationships. However, relying only on raw inputs in high-dimensional financial data may introduce redundancy and noise. This increases computational costs and reduces generalization. Without feature selection, the model becomes more sensitive to irrelevant or weak predictors, leading to instability. To address this, future work could utilize reinforcement learning–based feature selection or interpretability-driven approaches, such as SHAP and local interpretable model-agnostic explanations (LIME), to identify important variables. ML-ABC can also be combined with traditional methods, such as recursive feature elimination (REF), to optimize both parameter initialization and feature selection. This integration would improve robustness, reduce unnecessary complexity, and enhance fraud detection accuracy under real-world conditions.
- Fraud detection models risk bias if trained on unbalanced or non-representative data. For example, if transaction records overrepresent certain groups, the model may incorrectly flag their activities as fraudulent. Such bias can lead to significant ethical, financial, and legal consequences, particularly in the banking sector. The current model addresses class imbalance and improves minority-class detection. However, it does not include explicit fairness-mitigation strategies. Future work should explore fairness-aware methods such as reweighing, adversarial debiasing, or regularization techniques to reduce discrimination. Fairness metrics, such as demographic parity and equal opportunity, should also be included in the validation process. Continuous monitoring during deployment is crucial for detecting and correcting emergent biases. Clear reporting of training data and proactive bias management will help ensure the system supports fraud detection without unintended discrimination.
- The proposed model achieved strong results in controlled experiments. However, scalability and real-world deployment remain challenging. Fraud detection systems require very fast inference, as delays in transaction approval can result in significant financial losses. The model achieved an average decision-making time of about 18 milliseconds per transaction. This is suitable for many real-time applications, but it is slower than simpler models, such as XGBoost, revealing a trade-off between accuracy and computational cost. To address this, future work could employ model compression methods, such as pruning and quantization, to reduce computational demands without compromising accuracy. Deploying the model on scalable infrastructures, such as cloud-based GPU clusters or optimized edge devices, could also enhance performance. Another solution is to integrate it into a multi-stage detection system, where lightweight models handle obvious cases and the proposed hybrid method focuses on complex, high-risk transactions in real time.

# 5 Conclusions

This study presents a hybrid fraud detection framework that uses ML-ABC for weight initialization and PPO to handle class imbalance during ANN training. The contributions of this work are twofold: first, enhancing the stability and convergence of ANN training through optimized initialization, and second, improving minority class detection via adaptive reward shaping. Experimental results on the Université Libre de Bruxelles dataset demonstrate that the proposed model achieves 90.197% accuracy, 91.287% F-measure, and 89.456% G-means, surpassing the best existing benchmark by approximately 3%. These findings confirm a robust approach and demonstrate practical value for real-world e-commerce fraud detection, where accuracy and reliability are critical under severe imbalance.

In future work, we will extend the model to incorporate multi-modal data, transaction metadata, user behavior, and social media analytics, thereby capturing fraud patterns that are not visible in card data. Another direction is real-time detection, which requires faster optimization of PPO and ABC. Simplifying the network architecture or refining the ABC mutual learning process

can increase speed without reducing accuracy. These enhancements aim to build a robust, real-time fraud detection system for high-throughput e-commerce platforms.

## Appendix A

Table 8: Hyperparameters and their optimized values for metaheuristic algorithms.

| Algorithm | Hyperparameter | Value |
|---|---|---|
| HMS | Cognitive search factor (α) | 0.54 |
| | Memory size (M) | 25 |
| | Learning rate (α) | 0.71 |
| | Neighborhood size (L) | 15 |
| | Perturbation strength (σ) | 0.08 |
| | Learning rate (β) | 0.14 |
| FA | Light absorption coefficient (γ) | 1 |
| | Attractiveness at r = 0 (β0) | 0.62 |
| | Scaling factor (α) | 0.34 |
| BA | Constant for loudness update | 0.55 |
| | Constant for an emission rate update | 0.66 |
| | Initial pulse emission rate | 0.75 |
| DE | Scaling factor (FA) | 0.56 |
| | Crossover probability (CR) | 0.86 |
| | Differential weight (W) | 0.74 |
| | Scaling factor (FA) | 0.41 |
| GWO | Convergence factor (a) | 0.26 |
| COA | Discovery rate of alien solutions (pa) | 0.28 |
| | Step size (α) | 0.57 |
| | Levy flight exponent (λ) | 1.8 |
| AOA | Multiplication operator probability | 0.42 |
| | Division operator probability | 0.42 |
| | Subtraction operator probability | 0.45 |
| | Addition operator probability | 0.42 |
| PO | Exploration factor | 0.58 |
| | Exploitation factor | 0.51 |
| | Phase transition mechanism | 0.63 |
| SSA | Leader position update coefficient (c1) | 0.47 |
| | Random coefficient (c2) | 0.63 |
| ABC and ML-ABC | Number of bees | 62 |
| | Limit | 35 |
| | Number of cycles | 240 |
| | Colony size | 56 |
| | Probability of scout | 0.08 |
| | Number of elite bees | 4 |
| | Elite limit | 25 |

## Competing interests

The scholars claim no competing interests.

## Authorship contribution statement

Yuanyuan Zhang: Writing-Original draft preparation Conceptualization, Supervision, Project administration.

## Data availability

The scholars will make the raw data supporting this article's conclusions available without undue reservation.

## Declarations

Not applicable.

## Conflicts of interest

The scholars claimed no conflicts of interest considering this investigation.

## Author statement

The manuscript has been read and approved by all the authors, the requirements for authorship, as stated earlier

in this document, have been met, and each author believes that the manuscript displays honest work.

## Ethical approval

All scholars have been personally and actively involved in substantial work leading to the paper and will take public responsibility for its content.

# References


[1] Khalid, A.R., N. Owoh, O. Uthmani, M. Ashawa, J. Osamor and J. Adejoh (2024). Enhancing credit card fraud detection: an ensemble machine learning approach. *Big Data and Cognitive Computing*, MDPI, 8(1), https://doi.org/10.3390/bdcc8010006

[2] Gajakosh, A., R.A. Reddy and G. Rajalaxmi (2024). Fraud Detection in Credit Card Using Competitive Swarm Optimization with Support Vector Machine, In *2024 International Conference on Distributed Computing and Optimization Techniques (ICDCOT)*, IEEE, pp. 1–4. https://doi.org/10.1109/ICDCOT61034.2024.10515953

[3] Ida, S.J. and K. Balasubadra (2024). Enhancing credit card fraud detection through LSTM-based sequential analysis with early stopping, In *2024 2nd International Conference on Networking and Communications (ICNWC)*, IEEE, pp. 1–6. https://doi.org/10.1109/ICNWC60771.2024.10537550

[4] Botchey, F.E., Z. Qin, K. Hughes-Lartey and E.K. Ampomah (2022). Predicting fraud in mobile money transactions using machine learning: the effects of sampling techniques on the imbalanced dataset. *Informatica*, Slovene Society Informatika, 45(7). https://doi.org/10.31449/inf.v45i7.3179

[5] Liang, Z. and Y. Liang (2023). A study of identification of corporate financial fraud using neural network algorithms in an information-based environment. *Informatica*, Slovene Society Informatika, 47(9). https://doi.org/10.31449/inf.v47i9.5220

[6] Alatrista-Salas, H., J.F.A. Hancco and L. Espinoza-Villalobos (2025). Algorithms For Anomaly Detection on Time Series: A Use Case on Banking Data. *Informatica*, Slovene Society Informatika, 49(13). https://doi.org/10.31449/inf.v49i13.6243

[7] Rajput, B.S., P. Roy, S. Soni and B.S. Raghuwanshi (2024). ELM-based imbalanced data classification-A review. *Informatica*, Slovene Society Informatika, 48(2). https://doi.org/10.31449/inf.v48i2.5082

[8] Yıldız, B.S., S. Kumar, N. Panagant, P. Mehta, S.M. Sait, A.R. Yildiz, N. Pholdee, S. Bureerat and S. Mirjalili (2023). A novel hybrid arithmetic optimization algorithm for solving constrained optimization problems. *Knowledge-Based Systems*, Elsevier, 271, 110554. https://doi.org/10.1016/j.knosys.2023.110554

[9] Abdollahzadeh, B., N. Khodadadi, S. Barshandeh, P. Trojovský, F.S. Gharehchopogh, E.-S.M. El-kenawy, L. Abualigah and S. Mirjalili (2024). Puma optimizer (PO): a novel metaheuristic optimization algorithm and its application in machine learning. *Cluster Computing*, Springer, 27(4), pp. 5235–5283. https://doi.org/10.1007/s10586-023-04221-5

[10] Gharehchopogh, F.S., M. Namazi, L. Ebrahimi and B. Abdollahzadeh (2023). Advances in sparrow search algorithm: a comprehensive survey. *Archives of Computational Methods in Engineering*, Springer, 30(1), pp. 427–455. https://doi.org/10.1007/s11831-022-09804-w

[11] Kiani, F., S. Nematzadeh, F.A. Anka and M.A. Findikli (2023). Chaotic sand cat swarm optimization. *Mathematics*, MDPI, 11(10), p. 2340. https://doi.org/10.3390/math11102340

[12] Jafari, M., M.H.B. Chaleshtari, H. Khoramishad and H. Altenbach (2023). Minimization of thermal stress in perforated composite plate using metaheuristic algorithms WOA, SCA and GA. *Composite Structures*, Elsevier, 304, p. 116403. https://doi.org/10.1016/j.compstruct.2022.116403

[13] Kiani, F., F.A. Anka and F. Erenel (2023). PSCSO: Enhanced sand cat swarm optimization inspired by the political system to solve complex problems. *Advances in Engineering Software*, Elsevier, 178, p. 103423. https://doi.org/10.1016/j.advengsoft.2023.103423

[14] Mohsen, O.R., G. Nassreddine and M. Massoud (2023). Credit card fraud detector based on machine learning techniques. *Journal of Computer Science and Technology Studies*, 5(2), pp. 16–30. DOI: 10.32996/jcsts

[15] Golyeri, M., S. Celik, F. Bozyigit and D. Kılınç (2023). Fraud detection on e-commerce transactions using machine learning techniques. Artificial Intelligence *Theory and Applications*, 3(1), pp. 45–50.

[16] Sorour, S.E., K.M. AlBarrak, A.A. Abohany and A.A. Abd El-Mageed (2024). Credit card fraud detection using the brown bear optimization algorithm. *Alexandria Engineering Journal*, Elsevier, 104, pp. 171–192. https://doi.org/10.1016/j.aej.2024.06.040

[17] Halid, O.Y., B.T. Babalola, J. Sunday, S.O. Adejuwon, K.A. Adigun, O.F. Ogunboyo, T.O. Ogunlade, A.O. Ilesanmi and S.E. Fadugba (2024). On the Assessment of Trend and Pattern of COVID-19 Infection in Nigeria: Autoregressive Integrated Moving Average (ARIMA) Approach. *Pakistan Journal of Scientific & Industrial Research Series A: Physical Sciences*, 67(2), pp. 101–112.

[18] Krake, T., D. Klötzl, D. Hägele and D. Weiskopf (2024). Uncertainty-aware seasonal-trend

decomposition based on loess. *IEEE Transactions on Visualization and Computer Graphics*, IEEE, 31(2), pp. 1496–1512. https://doi.org/10.1109/TVCG.2024.3364388

[19] Suardiman, N., S. Dhanny, D. Tjahjadi, B. Permana and K. Ukar (2024). E-Commerce fraud detection using generated data from BANKSIM using machine learning approach: a pilot study, In *2024 18th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, IEEE, pp. 1–4. https://doi.org/10.1109/IMCOM60618.2024.10418372

[20] Odeyale, K.M., O.A. Moruff, S.I.T. Taofeekat and S.M. Kayode (2024). A support vector machine credit card fraud detection model based on high imbalance dataset. *Journal of Computers for Society*, 5(2), pp. 85–94.

[21] Umalwara, M., G. Aruna, V. Sushmalatha, M. Ruchinandan, P. Prathyusha, A. Thaseen and C. Padmaja (2024). Real time fraud detection credit card using ML approach, In *AIP Conf. Proc.*, AIP Publishing LLC, p. 020024. https://doi.org/10.1063/5.0195842

[22] Loukili, M., F. Messaoudi and H. Azirar (2024). E-Payment Fraud Detection in E-Commerce using Supervised Learning Algorithms, In *Advances in Emerging Financial Technology and Digital Money*, CRC Press, pp. 27–35.

[23] Abdul Salam, M., K.M. Fouad, D.L. Elbably and S.M. Elsayed (2024). Federated learning model for credit card fraud detection with data balancing techniques. *Neural Computing and Applications*, Springer, 36(11), pp. 6231–6256. https://doi.org/10.1007/s00521-023-09410-2

[24] Kokate, S. and M.S.R. Chettyi (2025). Fraudulent event detection in credit card data operations using SVM-RBF kernel function machine learning classification technique. *Intelligent Decision Technologies*, Sage Publications, 19(2), pp. 660–669. https://doi.org/10.1177/18724981241305118

[25] Wang, Y (2025). A Data Balancing and Ensemble Learning Approach for Credit Card Fraud Detection, In *2025 4th International Symposium on Computer Applications and Information Technology (ISCAIT)*, IEEE, pp. 386–390. https://doi.org/10.1109/ISCAIT64916.2025.11010591

[26] Baisholan, N., J.E. Dietz, S. Gnatyuk, M. Turdalyuly, E.T. Matson and K. Baisholanova (2025). FraudX AI: An Interpretable Machine Learning Framework for Credit Card Fraud Detection on Imbalanced Datasets. *Computers*, MDPI, 14(4), p. 120. https://doi.org/10.3390/computers14040120

[27] Xie, S., L. Liu, G. Sun, B. Pan, L. Lang and P. Guo (2023). Enhanced E-commerce Fraud Prediction Based on a Convolutional Neural Network Model. *Computers, Materials and Continua*, Elsevier, 75(1), pp. 1107–1117. https://doi.org/10.32604/cmc.2023.034917

[28] Karthika, J. and A. Senthilselvi (2023). An integration of deep learning model with Navo Minority Over-Sampling Technique to detect the frauds in credit cards. *Multimedia Tools and Applications*, Springer, 82(14), pp. 21757–21774. https://doi.org/10.1007/s11042-023-14365-6

[29] Al Balawi, S. and N. Aljohani (2023). Credit-card Fraud Detection System using Neural Networks. *International Arab Journal of Information Technology (IAJIT)*, 20(2). https://doi.org/10.34028/iajit/20/2/10

[30] Berhane, T., T. Melese, A. Walelign and A. Mohammed (2023). A Hybrid Convolutional Neural Network and Support Vector Machine-Based Credit Card Fraud Detection Model. *Mathematical Problems in Engineering*, Wiley Online Library, 2023(1), p. 8134627. https://doi.org/10.1155/2023/8134627

[31] Fanai, H. and H. Abbasimehr (2023). A novel combined approach based on deep Autoencoder and deep classifiers for credit card fraud detection. *Expert Systems with Applications*, Elsevier, 217, p. 119562. https://doi.org/10.1016/j.eswa.2023.119562

[32] Suganthi, V. and J. Jebathangam (2024). A Novel Approach for Credit Card Fraud Detection using Gated Recurrent Unit (GRU) Networks, In *2024 8th International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud)(I-SMAC)*, IEEE, pp. 1716–1721. https://doi.org/10.1109/I-SMAC61858.2024.10714795

[33] Mienye, I.D. and T.G. Swart (2024). A hybrid deep learning approach with generative adversarial network for credit card fraud detection. *Technologies*, MDPI, 12(10), p. 186. https://doi.org/10.3390/technologies12100186

[34] Onyeoma, C.F., H. Rafiq, D. Jeremiah, V.T. Ta and M. Usman (2024). Credit Card Fraud Detection Using Deep Neural Network with Shapley Additive Explanations, In *2024 International Conference on Frontiers of Information Technology (FIT)*, IEEE, pp. 1–6. https://doi.org/10.1109/FIT63703.2024.10838456

[35] Senthilselvi, A. and V. Nandhini (2024). A Novel Approach for Credit Card Fraud Detection Using Deep Learning Algorithms, In *2024 7th International Conference on Circuit Power and Computing Technologies (ICCPCT)*, IEEE, pp. 1870–1875. https://doi.org/10.1109/ICCPCT61902.2024.10672824

[36] Ganji, V.R. and A. Chaparala (2024). Wave Hedges distance-based feature fusion and hybrid optimization-enabled deep learning for cyber credit card fraud detection. *Knowledge and Information Systems*, Springer, 66(11), pp. 7005–7030. https://doi.org/10.1007/s10115-024-02177-5

[37] Veeru, B., N. Devender and K. Shoban (2024). Cryptographic Security in Credit Card Fraud Detection Using Homomorphic Encryption with CRO Based Hybrid BL-GRU Classification, In *Sustainable Development Using Private AI*, CRC Press, pp. 85–108. https://doi.org/10.3390/forecast7020031

[38] Yu, C., Y. Xu, J. Cao, Y. Zhang, Y. Jin and M. Zhu (2024). Credit card fraud detection using advanced transformer model, In *2024 IEEE International Conference on Metaverse Computing, Networking, and Applications (MetaCom)*, IEEE, pp. 343–350. https://doi.org/10.1109/MetaCom62920.2024.00064

[39] Dubey, P., P. Dubey and P.N. Bokoro (2025). A Unified Transformer–BDI Architecture for Financial Fraud Detection: Distributed Knowledge Transfer Across Diverse Datasets. *Forecasting*, MDPI, 7(2), p. 31. https://doi.org/10.3390/forecast7020031

[40] Kandi, K. and A. García-Dopico (2025). Enhancing Performance of Credit Card Model by Utilizing LSTM Networks and XGBoost Algorithms. *Machine Learning and Knowledge Extraction*, MDPI, 7(1), p. 20. https://doi.org/10.3390/make7010020

[41] Yousefimehr, B. and M. Ghatee (2025). A distribution-preserving method for resampling combined with LightGBM-LSTM for sequence-wise fraud detection in credit card transactions. *Expert Systems with Applications*, Elsevier, 262, p. 125661. https://doi.org/10.1016/j.eswa.2024.125661

[42] Khine, A.A. and Z.T.T. Myint (2025). Evaluating the Impact of Applied Sampling Algorithm on CNN and LSTM Models for Credit Card Fraud Analysis, In *2025 IEEE Conference on Computer Applications (ICCA)*, IEEE, pp. 1–6. https://doi.org/10.1109/ICCA65395.2025.11011102

[43] Sultana, I., S.M. Maheen, N. Kshetri and M.N.F. Zim (2025). detectGNN: Harnessing Graph Neural Networks for Enhanced Fraud Detection in Credit Card Transactions, In *2025 13th International Symposium on Digital Forensics and Security (ISDFS)*, IEEE, pp. 1–6. https://doi.org/10.1109/ISDFS65363.2025.11011957

[44] Zareeihemat, P., S. Mohamadi, J. Valipour and S.V. Moravvej (2025). Forecasting stock market volatility using housing market indicators: A reinforcement learning-based feature selection approach. *IEEE Access*, IEEE. https://doi.org/10.1109/ACCESS.2025.3554224