Journal of Industrial Intelligence
https://www.acadlore.com/journals/JII

Click Fraud Detection with Recurrent Neural Networks Optimized by Adapted Crayfish Optimization Algorithm

Lepa Babic, Vico Zeljkovic, Luka Jovanovic, Stefan Ivanovic, Aleksandar Djordjevic, Tamara Zivkovic, Miodrag Zivkovic, Nebojsa Bacanin*

Faculty of Informatics and Computing, Singidunum University, 11000 Belgrade, Serbia

* Correspondence: Nebojsa Bacanin (nbacanin@singidunum.ac.rs)

Received: 11-10-2024 **Revised:** 12-11-2024 **Accepted:** 12-16-2024

Citation: L. Babic, V. Zeljkovic, L. Jovanovic, S. Ivanovic, A. Djordjevic, T. Zivkovic, M. Zivkovic, N. Bacanin, "Click fraud detection with recurrent neural networks optimized by adapted crayfish optimization algorithm," *J. Ind Intell.*, vol. 2, no. 4, pp. 230–238, 2024. https://doi.org/10.56578/jii020405.

Abstract: Click fraud is a deceptive malicious strategy that relies on repetitive mimicking of human clicking on online advertisements, without actual intention to complete a purchase. This fraud can result in significant financial loses for both advertising companies and marketers, and at the same time destroying their public images. Nevertheless, detection of these illegitimate clicks is very challenging as they closely resemble to authentic human engagement. This study examines the utilization of artificial intelligence approaches to detect deceptive clicks, by identifying subtle correlations between the timing of the clicks, taking into account their geographical or network sources and linked application sources as indicators to separate legitimate from malicious activity. This study highlights the application of recurrent neural networks (RNNs) for this task, keeping in mind that the process of selection and tuning of the model's hyperparameters plays a vital role in the performance. An adapted implementation of crayfish optimization algorithm was consequently proposed in this paper, and used to optimize RNN models to enhance their general performance. The developed framework was evaluated utilizing actual operational datasets and yielded encouraging outcomes.

Keywords: Cybersecurity; Click fraud; Recurrent neural networks; Optimization; Hyperparameter tuning; Swarm intelligence; Crayfish optimization algorithm

1 Introduction

Online advertising has revolutionized how businesses engage with consumers by providing targeted and quantifiable promotional campaigns. Nevertheless, this digital approach has also given rise to advanced fraudulent activities, including click frauds [1, 2]. Click fraud is a malicious practice based on artificially synthesized fraudulent clicks aiming to affect advertising metrics and deplete advertising budgets. These fraudulent activities can be accomplished by utilizing automated bots, orchestrated click farms or even by competitors deliberately targeting to exhaust a rival's resources. The consequences of these practices can be devastating: key performance indicators like conversion rates, return on investment (ROI), and cost-per-click (CPC) are distorted, resulting in misinformed business strategies and possible considerable financial losses [3, 4].

The detection of click frauds can be particularly challenging because of the dynamic and evolving nature of fraudulent behavioral patterns. Traditional rule-based methods, which were utilized as a first line of defense through static IP block-lists and heuristic-based anomaly detection, are easily breached by fraudsters who continue to enhance and adapt their techniques to mimic genuine user behavioral patterns. In light of these obstacles, contemporary detection techniques have included application of artificial intelligence and deep learning, offering the capability to learn intricate patterns in immense volumes of data [5, 6].

In this study, a novel click frauds detection framework was proposed, relying on recurrent neural networks (RNNs) [7]. These networks are designated to analyze temporal click data and effectively differentiate among genuine and fraudulent actions by learning the underlying sequential patterns. To further improve the robustness and accuracy of the model, an adapted variant of swarm intelligence metaheuristics, in particular the crayfish optimization algorithm (COA) [8] was integrated into the suggested framework. This modified COA was tasked to automatically

fine-tune the hyperparameters of the RNN, ensuring that the network adapts promptly to emerging fraudulent patterns while maintaining high detection accuracy.

This research was consequently driven by three primary objectives:

- Development of a modified variant of COA algorithm, particularly devised to overcome the constraints of the baseline method and tailored for the click frauds detection challenge.
- Development of an RNN-based framework able to capture the intricate sequential dependencies linked with click frauds, consequently reducing false positives and enhancing the overall model reliability.
- Devised COA variant was integrated to this RNN-based security framework to perform hyperparameter optimization, with a goal to secure optimal performance for the problem in hand.

The rest of this manuscript is structured as follows: Section 2 yields a review of related literature on cybersecurity and click fraud detection. It also discusses hyperparameter optimization and the RNN architecture together with its role in analyzing sequential click data. Section 3 describes the elementary COA algorithm, and proposes an adapted version of this metaheuristics. Next, Section 4 outlines the experimental setup while Section 5 presents the experimental results. Ultimately, Section 6 discusses the implications of the findings and delineates future research directions.

2 Related Works

Click frauds detection has been a subject of active research over the years. Early approaches predominantly relied on rule-based systems, making use of static IP block-lists, heuristics, and threshold-based anomaly detection methods for identification of suspicious activities. Paper [9] explored the application of block-list driven firewall implementations to counter large-scale fraudulent activities, while authors [10] investigated IP-based classification mechanisms to filter out malicious traffic. Despite their initial success, these conventional approaches have several inherent constraints. As fraudsters adopted more sophisticated and adaptive techniques, static rules quickly became deprecated. More recent research published [11] have highlighted that these conventional methods frequently fail to scale properly and adapt to the nuanced attacks employed by modern click fraudsters.

The recent advances of deep learning have provided an adaptive alternative to traditional approaches, with RNNs emerging as particularly well-suited for processing sequential data. RNNs are capable of identifying temporal dependencies and modeling dynamic behavior found inside data, making them a natural choice for clickstream data analysis. Moreover, development of attention mechanisms has further refined the capabilities of RNNs by providing the models with capability to focus on the most salient parts of the input sequence [12]. Complementary studies like [13] and [14] have demonstrated that deep learning structures can greatly improve the identification of subtle behavioral patterns, consequently enhancing the overall performance of fraud detection systems.

However, the biggest obstacle in deployment of deep learning models in general is the optimization of hyperparameters. The performance of these models is highly sensitive to the configuration of their hyperparameters, which encompass learning rates, weight initialization schemes, and architecture-specific parameters. Manual tuning of these hyperparameters is not practical, and deterministic methods are not feasible since this is widely regarded as NP-hard problem. Metaheuristic approaches have emerged as a powerful solution to this problem. Among these, nature-inspired algorithms like particle swarm optimization (PSO) [15] and genetic algorithm (GA) [16] have been successfully applied to optimize model parameters. More recently, the crayfish optimization algorithm (COA) has shown promising results in navigating high-dimensional parameter spaces effectively [8], along with several other contemporary methods like red fox optimizer (RFO) [17], elk herd optimizer (EHO) [18] and salp swarm algorithm (SSA) [19].

The notable examples of successful hyperparapeters tuning by metaheuristics algorithms range over different application domains. These algorithms were used in medicine [20, 21], smart power grids [22], software development [23, 24], and sentiment analysis [25–27]. The field of cybersecurity also utilized this kind of approach, for intrusion detection [28, 29], insider threat detection [30], phishing detection [31], Metaverse and IoT networks in general [32].

In this research, a modified variant of COA was integrated into RNN framework, with a goal to optimize RNNs hyperparameters and enhance the accuracy and adaptability of click fraud detection systems.

2.1 Recurrent Neural Network (RNN)

Recurrent neural networks (RNNs) are a class of neural networks specifically designed for sequential data analysis. Unlike traditional feedforward networks, RNNs incorporate feedback loops that allow information to persist, thereby enabling the network to capture temporal dependencies. In many applications, such as natural language processing and time-series forecasting, RNNs have demonstrated the ability to learn complex patterns that evolve over time.

In this formulation, the RNN is designed to process clickstream data by maintaining a hidden state that encapsulates information from previous time steps. At each time step t, the RNN receives an input X_t and updates its hidden state h_t based on both the current input and the previous hidden state h_{t-1}. This update is performed using a nonlinear activation function, which in this case is the hyperbolic tangent function ($\tanh$). The update equation is given by:

$$h_t = \tanh(W_{xh} X_t + W_{hh} h_{t-1} + b_h) \tag{1}$$

Here, W_{xh} is the weight matrix associated with the input, W_{hh} is the recurrent weight matrix for the hidden state, and b_h is a bias term. This recurrent formulation allows the network to "remember" previous inputs, which is crucial for detecting patterns in click behavior that may span multiple time steps.

Once the hidden state is updated, the network generates an output Y_t through a linear transformation of the hidden state:

$$Y_t = W_{hy} h_t \tag{2}$$

In this equation, W_{hy} is the weight matrix that maps the hidden state to the output space. The simplicity of this architecture ensures that the RNN can be trained efficiently while still capturing the essential temporal features present in the clickstream data. Although attention mechanisms have been introduced in many recent works to further enhance performance, this papers focus in this section is on the fundamental RNN structure that forms the basis of the detection model.

3 Methods

This section first discloses the baseline implementation of COA metaheuristics. Afterwards, it points out the constraints of the original COA, and proposes an adapted variant that was later employed in the experiments which were carried out.

3.1 Crayfish Optimization Algorithm (COA)

Hyperparameter tuning is critical for the performance of deep learning structures, however, the high dimensionality of the parameter space and the dynamic nature of the problem domain often render traditional deterministics approaches inefficient and error-prone. The crayfish optimization algorithm (COA) [8]offers a robust, nature-inspired solution to this problem by mimicking the exploratory and social behaviors observed in crayfish.

Fundamental traits of metaheuristics algorithms encompass two primary stages: exploration and exploitation. A bio-inspired algorithm modeled after a tiny freshwater creature crayfish is referred to as COA. This organism is particularly attractive due to its intolerance of heat exceeding 30 degrees Celsius and its instinctive behavior for locating and fighting over shelters. Its competitive tendency become more exposed during the exploitation stage, as once a refuge is identified, that specific location may be claimed or found by another individual in the same time. The crayfish exits the refuge or sanctuary once the ambient temperature falls back below the 30 degrees mark, showcasing its adaptive response to environmental conditions.

While in exploration stage, the shelter is defined by Eq. (3).

$$X_{\text{shade}} = \frac{X_G + X_L}{2}, \tag{3}$$

here, the X_G depicts the optimal position over iterations, while the ideal position of the present collection of individuals is X_L. While the synthesized $rand$ number is lower than 0.5, a crayfish enters the cave without fighting over it as given by Eq. (4).

$$X_{i,j}^{t+1} = X_{i,j}^{t} + C_2 \times \text{rand} \times (X_{\text{shade}} - X_{i,j}^{t}), \tag{4}$$

here ,the t signifies the ongoing iteration, $t+1$ depicts the following one, and C_2 is computed with Eq. (5).

$$C_2 = 2 - \frac{t}{T}, \tag{5}$$

here, T represents the maximal count of rounds.

In case the $rand$ number is over 0.5, the Eq. (6) is used to model the competition among individuals.

$$X_{i,j}^{t+1} = X_{i,j}^{t} - X_{z,j}^{t} + X_{\text{shade}}, \tag{6}$$

here, z represents a random crayfish that is given by Eq. (7).

$$z = \text{round}(\text{rand} \times (N-1)) + 1. \tag{7}$$

The feeding phase commences when temperature falls below threshold of 30 degrees Celsius. The feeding supply is labeled by $X_{food} = X_G$, while the foraging behavior is outlined within Eq. (8).

$$X_{(13),\, t+1_{i,j}} = X_{t_{i,j}} + X_{\text{food}} \times p \times (\cos(2 \times \pi \times \text{rand}) - \sin(2 \times \pi \times \text{rand})). \tag{8}$$

In case the food supply is too big for the individual to eat it, it will break it into smaller pieces. The food size limit can be modeled by $Q \leq \frac{C3+1}{2}$, while the remainder of this task is provided by Eq. (9).

$$X_{\text{food}} = \exp\left(\frac{-1}{Q}\right) \times X_{\text{food}}, \tag{9}$$

here, the Q signifies the food size delineated with Eq. (10).

$$Q = C3 \times \text{rand} \times \left(\frac{\text{fitness}_i}{\text{fitness}_{\text{food}}}\right), \tag{10}$$

in this equation, the C_3 represents the biggest food supply with constant score 3, $fitness_i$ depicts the i-th crayfish's fitness score while the $fitness_{food}$ represents its location.

In case food supply is not too large, crayfish begins consuming it as depicted with Eq. (11).

$$X_{i,j}^{t+1} = (X_{i,j}^{t} - X_{\text{food}}) \times p + p \times \text{rand} \times X_{i,j}^{t}. \tag{11}$$

3.2 Modified COA Metaheuristics

Notwithstanding that COA is a novel algorithm that exhibited excellent performance in different domains, it still has the potential for improvement. More precisely, both exploration and exploitation stages of the baseline COA are the candidates for further improvement. Therefore, this study proposes an adaptive version of the algorithm that can enhance both aspects of COA.

In the first half of the execution (first $T/2$ iterations), the target is to booster the exploration. The individual having the poorest fitness score (the worst crayfish) is substituted by the fresh individual created as a hybrid between a pair of random crayfish units, by applying uniform crossover procedure inherited from genetic algorithm GA [16].

During latter $T/2$ rounds, the target is drifted toward exploitation. In this stage, the weakest crayfish in population is substituted by a hybrid between the most superior individual and a random crayfish, again by utilizing crossover operation. This adapted variation of COA is labeled adaptive COA (ACOA), while its pseudo-code is outlined by Algorithm 1.

Algorithm 1 ACOA metaheuristics pseudo-code

```
Produce starting population P of N random solutions
while (t < T) do
  for (every crayfish in P) do
    Utilize original COA search process
  end for
  Arrange individuals in P with respect to their fitness scores
  if (t < T/2) then
    Replace the poorest crayfish within P by a hybrid between a pair of arbitrary individuals, utilizing crossover mechanism.
  else
    Replace the poorest crayfish within P by a hybrid between the best crayfish and an arbitrary crayfish, utilizing crossover
    mechanism.
  end if
end while
return Crayfish with the best fitness score in P
```

Taking into account that this adaptation does not add any supplementary fitness function evaluations (FFE), that is regarded as the most processing heavy operation during metaheuristics execution, the suggested ACOA complexity is the same as the original COA with respect to FFEs.

4 Experimental Setup

This study employs dataset which is publicly available on Kaggle to evaluate introduced click fraud detection framework. Since this dataset is extremely large in the original format (having more than 11GB of samples), it was reduced to approximately 1% of original size to simplify and streamline the train and test tasks. Additionally, as the baseline dataset is highly imbalanced, where just around 1% of samples are linked to the real click fraud activities, models in this study were allocated with a balanced data consisting of 50% normal and 50% malicious samples to execute training procedure. This was achieved by the majority class undersampling strategy. The resulting balanced data was split into 70%/30% portions utilized for train and test activities for the observed models.

The performance of the introduced ACOA metaheuristic algorithm was evaluated by side by side comparisons to a set of powerful modern optimizers, encompassing baseline COA [8], GA [16], particle swarm optimization

(PSO) [15], Harris hawks optimization (HHO) algorithm [33], whale optimization algorithm (WOA) [34], reptile search algorithm (RSA) [35] and sinh cosh algorithm (SCHO) [36]. Contending algorithms were developed independently in Python, making use of default configurations of their control parameters as suggested by their respective creators. Every evaluated algorithm was assigned 8 solutions in populace and 8 iterations to execute optimization. Since metaheuristics algorithms inherit randomness factor from their stochastic nature, simulations were run in 30 separate executions. All regarded optimizers were given the task to enhance the models' performance through hyperparameter optimization. Table 1 outlines the collection of adjusted RNN parameters along with their search domains.

Table 1. RNN tuned parameters and their ranges.

Constraint	Learning Rate	Dropout	Learning Rate	Neurons within Layer	Count of Layers
Min	.0001	.05	5	32	1
Max	.0100	.20	10	128	2

Inside the model optimization stage, Matthews correlation coefficient (MCC) [37] was employed as the target optimization function for training the RNN architectures. The resulting structures were assessed utilizing a conventional suite of classifier's performance indicators across the conducted experiments [38]. This set of evaluation criteria included accuracy, precision, recall and the f1-score, as formally specified in Eqs. (12)–(15). Classification error rate was used as indicator through the simulations.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{12}$$

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

$$F1\_value = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{15}$$

5 Simulation Results

Table 2 outlines the outcomes for the objective function tuning experiments over 30 independent executions, where the finest score in each class is denoted in bold text. The suggested ACOA exhibited superior performance, by attaining the best outcomes for the best run, mean and median values of 0.571087, 0.568229 and 0.569089, respectively. In this scenario, HHO obtained the finest score in the worst execution, while GA exhibited superior stability of the outcomes across independent executions, having the smallest scores for standard deviation and variance among the regarded optimizers.

Table 2. Fitness function scores for optimized RNN models.

Method	Best	Worst	Mean	Median	Std	Var
RNN-ACOA	**0.571087**	0.564395	**0.568229**	**0.569089**	0.002519	6.35E-06
RNN-COA	0.567076	0.564786	0.566084	0.566671	0.001035	1.07E-06
RNN-GA	0.564996	0.562849	0.563754	0.563636	**0.000814**	**6.62E-07**
RNN-PSO	0.570810	0.563710	0.566789	0.565632	0.002591	6.72E-06
RNN-HHO	0.569627	**0.565489**	0.567465	0.566769	0.001681	2.83E-06
RNN-WOA	0.566591	0.562206	0.565235	0.565937	0.001562	2.44E-06
RNN-RSA	0.566549	0.561275	0.563912	0.564391	0.001896	3.59E-06
RNN-SCHO	0.565918	0.562831	0.564168	0.564241	0.001109	1.23E-06

Figure 1 showcases comparisons of the regarded optimizers' stability across separate executions. Exhibited violin diagram suggests that the suggested ACOA is not the most stable optimizer, as it is clearly surpassed by several other metaheuristics including GA, original COA and SCHO. However, despite these other methods performed more consistently, it did not help them secure the best overall score, which was attained by ACOA, suggesting their tendency to get stuck in local optimums easier than proposed algorithm. On the same Figure 1, the convergence graphs of the objective are also depicted, providing valuable information into the capabilities of regarded algorithms to handle falling into local optimums, and successfully converge to better regions of the search realm. It is obvious that the suggested ACOA established the finest overall solution during the fourth iteration of the best run, outperforming other regarded algorithms that were struggling to overcome local optimum pitfalls.

The working principle of the technique is shown in Figure 1.

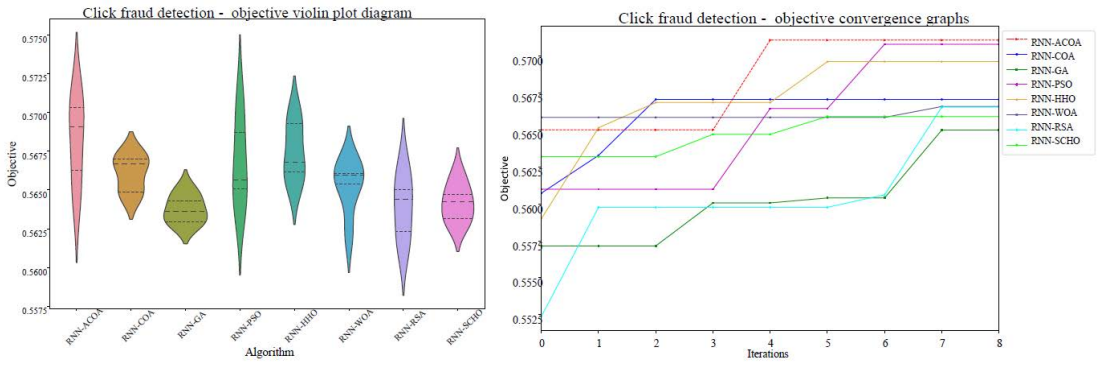

Figure 1. Objective function distribution and convergence diagrams

Table 3 depicts the outcomes of the indicator function (classification error). In this metric, SCHO obtained the finest overall outcome with error rate of 0.217883. Despite error rate was not established as the tuning target, introduced ACOA also attained respectable outcome, having the best score of 0.218978. Additionally, Figure 2 outlines the box plots and convergence graphs of the error rate.

Table 3. Indicator function (error) scores for optimized RNN models.

Method	Best	Worst	Mean	Median	Std	Var
RNN-ACOA	0.218978	0.219708	0.219051	0.218978	0.001092	1.19E-06
RNN-COA	0.220073	0.218978	0.218905	0.218978	0.000963	9.27E-07
RNN-GA	0.220438	0.220803	0.220511	0.220438	0.000273	7.46E-08
RNN-PSO	0.222628	0.220073	0.219270	0.218613	0.001978	3.91E-06
RNN-HHO	0.218613	0.218248	0.218613	0.218248	0.001199	1.44E-06
RNN-WOA	0.218248	0.223358	0.220073	0.218978	0.002165	4.69E-06
RNN-RSA	0.218613	0.221533	0.219635	0.218978	0.001068	1.14E-06
RNN-SCHO	0.217883	0.220438	0.219343	0.219708	0.000864	7.46E-07

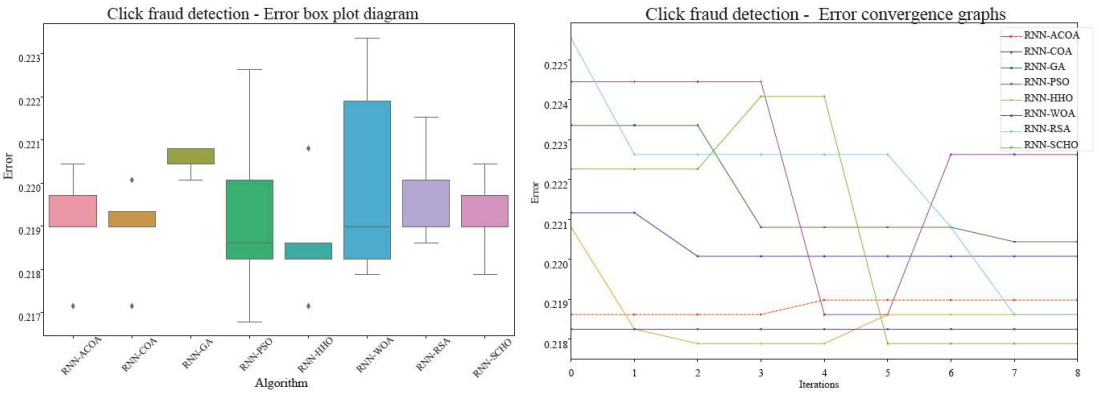

Figure 2. Error function distribution and convergence diagrams

Table 4 provides insight into the extensive comparative evaluation of the metrics for the best produced RNN models by every regarded optimizer. The introduced ACOA generated RNN structure that obtained excellent accuracy of 0.781022, accompanied by good overall values of macro and weighted averages, and balanced per class precision, recall and f1-score. It is also obvious that other optimizers also produced excellent RNN structures, with less than 1% differences in accuracy between the finest produced RNNs by each metaheuristics.

Figure 3 shows supplementary visualizations of the attained outcomes in the form of the best ACOA produced RNN's confusion matrix and ROC diagram. Finally, the hyperparameter selections for the best produced RNNs by every regarded optimizer are summarized within Table 5, to facilitate possible future replications of these simulations and support the consistency of the experimental outcomes. The configurations shown in Table 5 achieved the scores presented in Table 4.

Table 4. Best performing model detailed metric comparisons.

Method	Metric	Normal	Click-fraud	Accuracy	Macro Avg	Weighted Avg
RNN-ACOA	precision	0.738095	0.840909	0.781022	0.789502	0.789877
	recall	0.866176	0.697101	0.781022	0.781639	0.781022
	f1-score	0.797023	0.762282	0.781022	0.779653	0.779526
RNN-COA	precision	0.740929	0.832335	0.779927	0.786632	0.786966
	recall	0.855882	0.705072	0.779927	0.780477	0.779927
	f1-score	0.794268	0.763437	0.779927	0.778852	0.778740
RNN-GA	precision	0.743871	0.826050	0.779562	0.784961	0.785261
	recall	0.847794	0.712319	0.779562	0.780056	0.779562
	f1-score	0.792440	0.764981	0.779562	0.778710	0.778610
RNN-PSO	precision	0.723747	0.861842	0.777372	0.792795	0.793299
	recall	0.891912	0.664493	0.777372	0.778202	0.777372
	f1-score	0.799078	0.750409	0.777372	0.774743	0.774566
RNN-HHO	precision	0.743131	0.832340	0.781387	0.787736	0.788061
	recall	0.855147	0.708696	0.781387	0.781921	0.781387
	f1-score	0.795214	0.765558	0.781387	0.780386	0.780277
RNN-WOA	precision	0.754679	0.814309	0.781752	0.784494	0.784712
	recall	0.830147	0.734058	0.781752	0.782103	0.781752
	f1-score	0.790616	0.772104	0.781752	0.781360	0.781292
RNN-RSA	precision	0.751820	0.817738	0.781387	0.784779	0.785020
	recall	0.835294	0.728261	0.781387	0.781777	0.781387
	f1-score	0.791362	0.770410	0.781387	0.780886	0.780810
RNN-SCHO	precision	0.761480	0.805621	0.782117	0.783551	0.783712
	recall	0.816912	0.747826	0.782117	0.782369	0.782117
	f1-score	0.788223	0.775648	0.782117	0.781936	0.781890
	support	1360	1380			

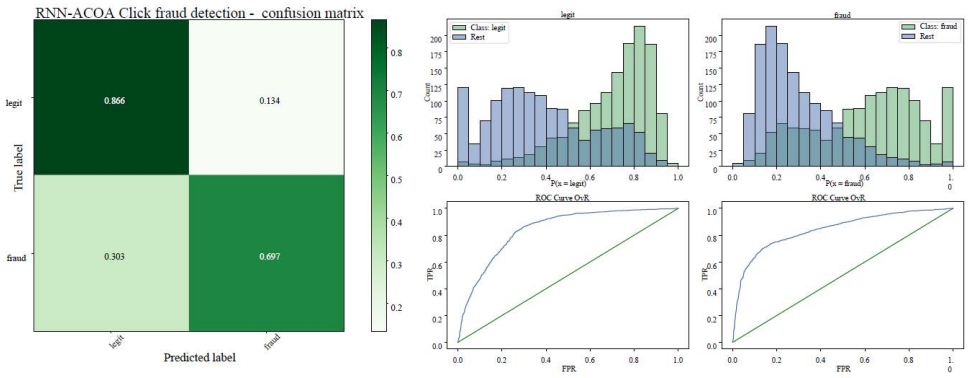

Figure 3. Best model confusion matrix and ROC plot

Table 5. Parameter selection for best performing RNN models.

Method	Learning rate	Dropout	Training epochs	RNN Layers	Layer 1 Neurons	Layer 2 Neurons
RNN-ACOA	1.00e-02	2.00e-01	10	2	93	55
RNN-COA	1.00e-02	5.00e-02	10	2	108	110
RNN-GA	9.10e-03	5.85e-02	8	2	32	71
RNN-PSO	1.00e-02	1.58e-01	6	2	73	117
RNN-HHO	8.09e-03	1.18e-01	9	2	35	96
RNN-WOA	1.00e-02	2.00e-01	10	2	128	83
RNN-RSA	9.26e-03	7.88e-02	10	2	64	65
RNN-SCHO	3.21e-03	1.98e-01	10	2	32	128

6 Conclusion

In the domain of digital advertisement business, click frauds persist as a critical obstacle, frequently resulting in significant financial losses for advertisers and compromising the reputation of ad-serving companies. The crucial task of reliable separation of legitimate human interactions from fraudulent click activities requires implementation of sophisticated detection frameworks. This research investigated the efficiency of AI-driven approaches, particularly those utilizing RNN architectures, to identify anomalous click patterns that indicate malicious behavior. To improve the predictive capabilities of RNN models, an adapted version of the COA algorithm was implemented to optimize RNN hyperparameter configuration. This introduced framework was evaluated using real-world data and demonstrated promising results, with the most effective models achieving accuracy rates of up to 78.2%.

Despite promising outcomes, this study was not without several constraints. The large volume of available data imposed limits to the amount of data that could be feasibly employed for model training and testing activities. Consequently, down-sampled dataset was used, that could have marginally affected overall performance results. Additionally, the computational intensity linked to optimization tasks restricted both the population sizes and iterations permitted to the applied optimization algorithms.

Future research will aim to address these constraints by exploring more scalable data handling strategies and improving computational resources efficiency. Supplementary investigations will target broader applications of the suggested ACOA and the integration of hybrid approaches to further advance the accuracy and reliability of click frauds detection systems.

Acknowledgements

This research was supported by the Science Fund of the Republic of Serbia, grant No. 7373, characterizing crises-caused air pollution alternations using an artificial intelligence-based framework (crAIRsis), and grant No. 7502, Intelligent Multi-Agent Control and Optimization applied to Green Buildings and Environmental Monitoring Drone Swarms (ECOSwarm).

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] K. R. Dörnyei, "Marketing professionals' views on online advertising fraud," *J. Current Issues Res. Adv.*, vol. 42, no. 2, pp. 156–174, 2020. https://doi.org/10.1080/10641734.2020.1737276

[2] R. Jamil Alzghoul, E. Emad Abdallah, and S. Abdel-hafiz Al-khawaldeh, "Fraud in online classified Ads: Strategies, risks, and detection methods: A survey," *J. Appl. Sec. Res.*, vol. 19, no. 1, pp. 45–69, 2022. https://doi.org/10.1080/19361610.2022.2124328

[3] S. Sadeghpour and N. Vlajic, "Click fraud in digital advertising: A comprehensive survey," *Computers*, vol. 10, no. 12, p. 164, 2021. https://doi.org/10.3390/computers10120164

[4] S. Sadeghpour and N. Vlajic, "Ads and fraud: A comprehensive survey of fraud in online advertising," *J. Cyb. Privacy*, vol. 1, no. 4, pp. 804–832, 2021. https://doi.org/10.3390/jcp1040039

[5] A. Reem Alzahrani and M. Aljabri, "Ai-based techniques for Ad click fraud detection and prevention: Review and research directions," *J. Sensor Actuator Net.*, vol. 12, no. 1, p. 4, 2022. https://doi.org/10.3390/jsan12010004

[6] D. Sisodia and D. S. Sisodia, "A transfer learning framework towards identifying behavioral changes of fraudulent publishers in pay-per-click model of online advertising for click fraud detection," *Expert Sys. Appli.*, vol. 232, p. 120922, 2023. https://doi.org/10.1016/j.eswa.2023.120922

[7] L. Medsker and L. C. Jain, "Recurrent Neural Networks: Design and Applications," 1999.

[8] H. Jia, H. Rao, C. Wen, and S. Mirjalili, "Crayfish optimization algorithm," *Artif. Intell. Rev.*, vol. 56, no. 2, pp. 1919–1979, 2023. https://doi.org/10.1007/s10462-023-10567-4

[9] J. Kylmanen, "Information security improving blocklist driven firewall implementation," 2013. https://oulurepo.oulu.fi/handle/10024/38452

[10] D. Chiba, K. Tobe, T. Mori, and S. Goto, "Detecting Malicious Websites by Learning IP Address Features," in *2012 IEEE/IPSJ 12th International Symposium on Applications and the Internet*. IEEE, 2012, pp. 29–39. https://doi.org/10.1109/saint.2012.14

[11] H. Dao, J. Mazel, and K. Fukuda, "Understanding abusive web resources," in *Proceedings of the Asian Internet Engineering Conference*, 2018, pp. 54–61. https://doi.org/10.1145/3289166.3289174

[12] A. Vaswani, *Attention is all you Need*. Advances in Neural Information Processing Systems, 2017.

[13] C. Liu, Y. Gao, L. Sun, J. Feng, H. Yang, and X. Ao, "User behavior pre-training for online fraud detection," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025, pp. 3357–3365. http://baltijapublishing.lv/omp/index.php/bp/catalog/download/544/14659/30766-1

[14] J. Yang, S. Rahardja, and S. Rahardja, "Click fraud detection: Hk-index for feature extraction from variable-length time series of user behavior," X. 2022 IEEE 32nd International Workshop on Machine Learning for Signal Processing (MLSP), Ed., 2025, pp. 1–6. https://doi.org/10.1109/MLSP55214.2022.9943422

[15] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, 1995, pp. 1942–1948. https://doi.org/10.1109/icnn.1995.488968

[16] S. Mirjalili, *Genetic Algorithm*. Springer International Publishing, 2019. http://doi.org/10.1007/978-3-319-93025-1

[17] D. Polap and M. Wozniak, "Red fox optimization algorithm," *Exp. Sys. App.*, vol. 166, p. 114107, 2021.

[18] M. A. Al-Betar, M. A. Awadallah, M. S. Braik, S. Makhadmeh, and I. A. Doush, "Elk herd optimizer: a novel nature-inspired metaheuristic algorithm," *Artificial Intelligence Review*, vol. 57, no. 3, 2024. https://doi.org/10.1007/s10462-023-10680-4

[19] S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, and S. M. Mirjalili, "Salp swarm algorithm: A bio-inspired optimizer for engineering design problems," *Adv. Eng. Software*, vol. 114, pp. 163–191, 2017.

[20] L. Jovanovic, N. Bacanin, M. Zivkovic, M. Antonijevic, A. Petrovic, and T. Zivkovic, "Anomaly detection in ECG using recurrent networks optimized by modified metaheuristic algorithm," in *2023 31st Telecommunications Forum (TELFOR), Belgrade, Serbia*, 2023, pp. 1–4. https://doi.org/10.1109/telfor59449.2023.10372802

[21] A. Albakri and Y. M. Alqahtani, "Internet of medical things with a blockchain-assisted smart healthcare system using metaheuristics with a deep learning model," *App. Sci.*, vol. 13, no. 10, p. 6108, 2023. https://doi.org/10.3390/app13106108

[22] M. Papadimitrakis, N. Giamarelos, M. Stogiannos, E. Zois, N. N.A.-I. Livanos, and A. Alexandridis, "Metaheuristic search in smart grid: A review with emphasis on planning, scheduling and power flow optimization applications," *Renew. Sus. Energy Rev.*, vol. 145, p. 111072, 2021. https://doi.org/10.1016/j.rser.2021.111072

[23] M. S. Khan, F. Jabeen, S. Ghouzali, Z. Rehman, S. Naz, and W. Abdul, "Metaheuristic algorithms in optimizing deep neural network model for software effort estimation," *IEEE Access*, vol. 9, pp. 60 309–60 327, 2021. https://doi.org/10.1109/access.2021.3072380

[24] T. Zivkovic, B. Nikolic, V. Simic, D. Pamucar, and N. Bacanin, "Software defects prediction by metaheuristics 281 tuned extreme gradient boosting and analysis based on shapley additive explanations," *Applied Soft Computing*, vol. 146, p. 110659, 2025.

[25] M. K. Suddle and M. Bashir, "Metaheuristics based long short term memory optimization for sentiment analysis," *App. Soft Com.*, vol. 131, p. 109794, 2022. https://doi.org/10.1016/j.asoc.2022.109794

[26] M. Dobrojevic, L. Jovanovic, L. Babic, M. Cajic, T. Zivkovic, M. Zivkovic, S. Muthusamy, M. Antonijevic, and N. Bacanin, "Cyberbullying sexism harassment identification by metaheurustics-tuned extreme gradient boosting," *Com. Mat. Continua*, vol. 80, no. 3, pp. 4997–5027, 2024. https://doi.org/10.32604/cmc.2024.054459

[27] L. Babic, L. Jovanovic, A. Petrovic, M. Zivkovic, T. Zivkovic, and N. Bacanin, "Leveraging metaheuristic optimized machine learning classifiers to determine employee satisfaction," in *International Conference on 290 Multi-Strategy Learning Environment*, pp. 337–352.

[28] P. Dakic, M. Zivkovic, L. Jovanovic, N. Bacanin, M. Antonijevic, J. Kaljevic, and V. Simic, "Intrusion detection using metaheuristic optimization within iot/iiot systems and software of autonomous vehicles," *Sci. Rep.*, vol. 14, no. 1, p. 22884, 2024.

[29] S. M. T. Nizamudeen, "Intelligent intrusion detection framework for multi-clouds – IoT environment using swarm-based deep learning classifier," *Journal of Cloud Computing*, vol. 12, no. 1, p. 134, 2023. https://doi.org/10.1186/s13677-023-00509-4

[30] D. Mladenovic, M. Antonijevic, L. Jovanovic, V. Simic, M. Zivkovic, N. Bacanin, T. Zivkovic, and J. Perisic, "Sentiment classification for insider threat identification using metaheuristic optimized machine learning classifiers," *Sci. Rep.*, vol. 14, no. 1, p. 25731, 2024.

[31] B. Lakicevic, Z. Spalevic, I. Volas, L. Jovanovic, M. Zivkovic, T. Zivkovic, and N. Bacanin, *Artificial neural networks with soft attention: Natural language processing for phishing email detection optimized with modified metaheuristics*, 2024, pp. 421–438. https://doi.org/10.1007/978-3-031-83790-6_27

[32] A. Sagu, N. S. Gill, P. Gulia, P. K. Singh, and W. Hong, "Design of metaheuristic optimization algorithms for deep learning model for secure IoT environment," *Sustainability*, vol. 15, no. 3, p. 2204, 2023. https://doi.org/10.3390/su15032204

[33] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: 306 Algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872.

[34] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Soft.*, vol. 95, pp. 51–67, 2016.

[35] L. Abualigah, M. A. Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile search algorithm (rsa): A nature-inspired meta-heuristic optimizer," *Expert Systems with Applications*, vol. 191, p. 116158, 2022.

[36] J. Bai, Y. Li, M. Zheng, S. Khatir, B. Benaissa, L. Abualigah, and M. Abdel Wahab, "A sinh cosh optimizer," *Knowle. Sys.*, vol. 282, p. 111081, 2023.

[37] B. Matthews, "Comparison of the predicted and observed secondary structure of T4 phage lysozyme," *Biochi. Bioph. Acta (BBA)*, vol. 405, no. 2, pp. 442–451, 1975. https://doi.org/10.1016/0005-2795(75)90109-9

[38] M. Hossin and M. N. Sulaiman, "A review on evaluation metrics for data classification evaluations," *Int. J. Data Min. Knowle. Mana. Proc.*, vol. 5, no. 2, 2015.