NSS

Neutrosophic Sets and Systems, Vol. 82, 2025

University of New Mexico

Enhanced Federated Learning Framework based on Deep Learning and Neutrosophic Set for Android Malware Classification

Mohamed Refaat Abdellah [1], Hasan H. Oudah[2], Ahmed Mohamed Ahmed Badawy [3]*, Mohamed AbdElFattah AbdElFattah M Hassan[4], Shady Ahmed Bedier[5], Ahmed A. Metwaly[6], Mohamed eassa[5,7], Ahmed Abdelhafeez[5,7]

[1]The Department of Computer Science, College of Information Technology, Misr University for Science and Technology, Cairo, Egypt, mohamed.refaat@must.edu.eg

[2]College of Arts / Department of Journalism, Ahl Al Bayt University, Karbala, Iraq, hasan.hadi@abu.edu.iq

[3]Cybersecurity Department, Faculty of Computers and Artificial Intelligence, Helwan University, Egypt; amb_eg@hotmail.com

[4]Cybersecurity Department, Faculty of Computers and Artificial Intelligence, Helwan University, Egypt; Eng.Mohamed.se20@gmail.com

[5]Computer Science Department, Faculty of Information System and Computer Science, October 6 University, Giza, 12585, Egypt; Shady.Bedier.CS@o6u.edu.eg; mohamed.eassa.cs@o6u.edu.eg; aahafeez.scis@o6u.edu.eg

[6]Department of Computer Science, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt, a.metwaly23@fci.zu.edu.eg

[7]Applied Science Research Center. Applied Science Private University, Amman, Jordan
* Corresponding author's Email: amb_eg@hotmail.com

Abstract: Malware detection is one of the critical tasks of cybersecurity, especially considering the growing popularity of mobile devices. The integrity and security of mobile ecosystems rely on the capacity to identify malware quickly and precisely, as security hacks against these devices become increasingly frequent. Due to the requirement for massive volumes of data aggregation, traditional centralized machine learning (ML) techniques for malware detection face challenges with data sharing, computational complexity, and privacy. This research addresses these challenges by proposing a novel model, called "CNN-Fed", which is based on Convolutional Neural Networks (CNN) with Federated Learning (Fed). The main goal of this work is to create a global classifier for Android malware detection that is highly accurate and does not require centralised data aggregation. Using four benchmark datasets—Drebin, Malgenome, Kronodroid, and Tuandromd—the CNN-Fed model is trained across many clients in a federated setting under the suggested architecture. Then we use multi-criteria decision-making (MCDM) methodology to evaluate the number of clients and number of rounds. We use two MCDM methods such as Entropy methodology to compute the criteria weights and the CoCoSo methodology to rank the alternatives. We use the interval-valued neutrosophic sets (IVNSs) to deal with uncertainty and vague information. We use five criteria such as accuracy, F1-score, precision, recall, and FP and six of alternatives refer to the number of clients and number of rounds. The results show the best alternative has a larger number of clients and rounds.

Keywords: Malware detection; Deep Learning; Federated Learning; Interval Valued Neutrosophic Sets.

1. Introduction

Data security and user privacy are becoming increasingly of a problem with the proliferation of personal and large computing systems. Deep learning and conventional machine learning methods become less useful

as malware threats rise. Malware is a type of software that targets and steals important data from computer systems, digital systems, and mobile devices. The journey of malware began in 1986 with the development of the first malware, the brain[1], [2].

1.1 Malware

For cyber security services, malware consultation is an essential topic as one of malicious software, requiring a large investment of time and human resources. Still, there is a gap between recent research and practical implementations. Depending on the type, target, and attacker's target, malware can affect individuals, groups, companies, governments, and society. Common effects of malware include [3]:

- Data Theft and Breaches.
- Financial Loss.
- System Disruption.
- Data Destruction.
- Privacy Violations.
- Propagation.
- Reputation Damage.
- Resource Consumption.
- Supply Chain Attack.

The primary reason malware detection is crucial is that it shields users, networks, and computer systems from the damaging effects of malicious software. Here are some key arguments supporting the importance of malware detection:

- Security.
- Data Protection.
- System Integrity.
- Financial Impact.
- Privacy.
- Mitigating Reputation Damage.
- Preventing Secondary Infections.
- Protection against Evolving Threats.

There are three common techniques employed in malware analysis [3]:
Static Analysis: This approach involves recognizing malware signatures and extracting important information from codes or files. It includes establishing the identity of the malware, associated libraries, URLs and the programming language.
Dynamic Analysis: The method looks at how a certain type of malware behaves when it is active in the real world under different environmental conditions. This helps in understanding what such software is designed for, how it propagates and potential harm that may be caused.
Hybrid Malware Analysis: Combining both static and dynamic analyses, this method offers a more complete and exact assessment of the behaviour of malware with respect to its behaviour, capabilities, effects.

1.2 Interval valued neutrosophic sets (IVNSs)

Well-known techniques for handling complicated situations include multicriteria decision-making (MCDM) models, which rank, prioritize, and sort various options while considering several criteria. Decision-makers can use MCDM models to convey ambiguous viewpoints using a variety of uncertainty sets, such as fuzzy logic[4], [5].

In Boolean logic, a proposition can be true or false, but nothing in between; in optimization, a solution is either feasible or not; and in conventional set theory, an element can either belong to a set or not. However, practically everything in actual life is imprecise, a question of degree, and outside the scope of conventional reasoning.

Zadeh developed the fuzzy sets theory to address this type of ambiguity. It was initially presented in 1965 and has since been expanded to include several fuzzy set variations. Zadeh created the type-n fuzzy set to deal with the membership function's uncertainty in fuzzy set theory. Zadeh invented interval-valued fuzzy sets (IVFSs)[6], [7].

To properly characterize a membership function with its membership and non-membership degrees, including the hesitation of decision-makers, Atanassov introduced intuitionistic fuzzy sets (IFSs). Torra first defined hesitant fuzzy sets (HFSs), which are expansions of conventional fuzzy sets in which a single element may belong to a set of values. Even with all these additions, the description of membership functions could not accurately represent the sentiments of decision-makers. As a result, certain extensions are still needed.

To overcome this shortcoming, Smarandache extended intuitionistic fuzzy sets to create neutrosophic logic and neutrosophic sets (NSs)[8], [9]. Every element of the cosmos has a degree of truthiness, indeterminacy, and falsehood inside the nonstandard unit interval, which is known as the neutrosophic set. The degree of belongingness or non-belongingness and the indeterminacy value were the factors integrated as relativity or absoluteness in the neutrosophic sets, where uncertainty is represented as truth and falsity values. In addition to managing system uncertainty, neutrosophic sets also lessen the indecision of contradicting data using this notation[10]. As a result, we may independently think of the indeterminacy value as the hesitation degree and the truth and falsehood values as membership and non-membership degrees[11], [12].

1.3 Learning Techniques

1.3.1 Deep learning for malware detection

Convolutional Neural Networks, sometimes known as CNNs, have recently been suggested as a DL technique for detecting malware. These ideas most certainly do not, however, uphold the DL principle that is essential for detecting malware. Instead, a sequence of linear transformations is frequently suggested, which makes them difficult to learn efficiently because they may result in limited performance from DL or increase classification efficiency in the absence of training data. The challenges prompted a development in Malware detection techniques. Malware detection has used a variety of DL models. On the other hand, few deep-learning options are available for real-time virus detection in an operational system [13], [14].

1.3.2 Federated Learning

Google introduced Federated Learning (FL) in 2016, providing a workaround for data privacy concerns while facilitating cross-device collaborative learning. With this novel approach, devices can work together to train ML algorithms without directly exchanging sensitive raw data. Particularly around cybersecurity, FL has demonstrated potential as a malware detection technique. FL distributed architecture not only safeguards security and privacy on distributed networks but also effectively thwarts malware threats. Here, FL protects sensitive information by allowing local clients on various devices to build models using their data. These models are then combined by a global server, enabling the construction of an all-encompassing defense system against dynamic malware attacks[14].

This paper has the following contributions:

- CNN-FED, a new model: This model combines Deep Learning (DL) and Federated Learning (FL) algorithms to detect malware.
- Global classifier of high accuracy: CNN-FED has been designed in such a way that it can generate highly accurate global classifiers without having access to distributed data.
- Results from experiments: Four well-known datasets, Drebin, Malgenome, Kronodroid, and Tuandromd were used for experiments. The experimental findings show high levels of accuracy for CNN-FED.
- We use interval-valued neutrosophic sets (IVNSs) methodology to evaluate the number of clients with the number of rounds after obtaining the results of the DL models.

- We use two MCDM methods such as the Entropy method to compute the criteria weights and the CoCoSo method to rank the alternatives.

2. Methodology

This section shows the steps of the proposed approach. We show some definitions of the IVNSs[15], [16].

Definition 1.

The neutrosophic sets has three membership functions such as Truth $T_C(y)$, Indeterminacy $I_C(y)$, and Falsity $F_C(y)$.

$$0-\leq \sup T_C(y) + \sup T_C(y) + \sup T_C(y) \leq 3 + \tag{1}$$

Definition 2.

There are two intervals valued neutrosophic numbers (IVNNs) and in this part, we introduce their operations such as:

$$C + D = \begin{pmatrix} \begin{bmatrix} T_C^L(y) + T_D^L(y) - T_C^L(y)T_D^L(y), \\ T_C^U(y) + T_D^U(y) - T_C^U(y)T_D^U(y) \end{bmatrix}, \\ [I_C^L(y)I_D^L(y), I_C^U(y)I_D^U(y)], \\ [F_C^L(y)F_D^L(y), F_C^U(y)F_D^U(y)] \end{pmatrix} \tag{2}$$

$$CD = \begin{pmatrix} [T_C^L(y)T_D^L(y), T_C^U(y)T_D^U(y)], \\ \begin{bmatrix} I_C^L(y) + I_D^L(y) - I_C^L(y)I_D^L(y) \\ , I_C^U(y) + I_D^U(y) - I_C^U(y)I_D^U(y) \end{bmatrix}, \\ \begin{bmatrix} F_C^L(y) + F_D^L(y) - F_C^L(y)F_D^L(y), \\ F_C^U(y) + F_D^U(y) - F_C^U(y)T_D^U(y) \end{bmatrix} \end{pmatrix} \tag{3}$$

$$C - D = \begin{pmatrix} \begin{bmatrix} T_C^L(y) - T_D^U(y), \\ T_C^U(y) - T_D^L(y) \end{bmatrix}, \\ \begin{bmatrix} \max\left(I_C^L(y), I_D^L(y)\right), \\ \max\left(I_C^U(y), I_D^U(y)\right) \end{bmatrix}, \\ \begin{bmatrix} F_C^L(y) - F_D^U(y), \\ F_C^U(y) - F_D^L(y) \end{bmatrix} \end{pmatrix} \tag{4}$$

$$\frac{C}{\aleph} = \begin{pmatrix} \left[\min\left(\frac{T_C^L(y)}{\aleph}, 1\right), \min\left(\frac{T_C^U(y)}{\aleph}, 1\right)\right], \\ \left[\min\left(\frac{I_C^L(y)}{\aleph}, 1\right), \min\left(\frac{I_C^U(y)}{\aleph}, 1\right)\right] \\ \left[\min\left(\frac{F_C^L(y)}{\aleph}, 1\right), \min\left(\frac{F_C^U(y)}{\aleph}, 1\right)\right] \end{pmatrix} \tag{5}$$

$$\frac{C}{D}=\left(\begin{bmatrix}\min\left\{\frac{T_C^L(y)}{T_D^L(y)},\frac{T_C^L(y)}{T_D^U(y)},\\ \frac{T_C^U(y)}{T_D^L(y)},\frac{T_C^U(y)}{T_D^U(y)}\end{bmatrix},\\ \begin{bmatrix}\max\left\{\frac{T_C^L(y)}{T_D^L(y)},\frac{T_C^L(y)}{T_D^U(y)},\\ \frac{T_A^U(x)}{T_B^L(x)},\frac{T_A^U(x)}{T_B^U(x)}\end{bmatrix},\\ \begin{bmatrix}\min\left\{\frac{I_C^L(y)}{I_D^L(y)},\frac{I_C^L(y)}{I_D^U(y)}\right\},\\ \frac{I_C^U(y)}{I_D^L(y)},\frac{I_C^U(y)}{I_D^U(y)}\end{bmatrix},\\ \begin{bmatrix}\max\left\{\frac{I_C^L(y)}{I_D^L(y)},\frac{I_C^L(y)}{I_D^U(y)}\right\},\\ \frac{I_C^U(y)}{I_D^L(y)},\frac{I_C^U(y)}{I_D^U(y)}\end{bmatrix},\\ \begin{bmatrix}\min\left\{\frac{F_C^L(y)}{F_D^L(y)},\frac{F_C^L(y)}{F_D^U(y)}\right\},\\ \frac{F_C^U(y)}{F_D^L(y)},\frac{F_C^U(y)}{F_D^U(y)}\end{bmatrix},\\ \begin{bmatrix}\max\left\{\frac{F_C^L(y)}{F_D^L(y)},\frac{F_C^L(y)}{F_D^U(y)}\right\},\\ \frac{F_C^U(y)}{F_D^L(y)},\frac{F_C^U(y)}{F_D^U(y)}\end{bmatrix}\end{pmatrix} \tag{6}$$

$$C^{-1}=\begin{pmatrix}\left[\left(T_C^L(y)\right)^{-1},\left(T_C^U(y)\right)^{-1}\right],\\ \left[\left(I_C^L(y)\right)^{-1},\left(I_C^U(y)\right)^{-1}\right],\\ \left[\left(F_C^L(y)\right)^{-1},\left(F_C^U(y)\right)^{-1}\right]\end{pmatrix} \tag{7}$$

Definition 3.

We can compute the score function such as:

$$S(C)=\begin{pmatrix}\frac{T_C^L(y)+T_C^U(y)}{2}+\\ \left(1-\frac{\left(I_C^L(y)+I_C^U(y)\right)}{2}\right)\left(I_C^U(y)\right)-\\ \left(\frac{\left(F_C^L(y)+F_C^U(y)\right)}{2}\right)\left(1-F_C^U(y)\right)\end{pmatrix} \tag{8}$$

IVN-Entropy

This method is used to compute the criteria weights. The steps of the Entropy method are shown as follows[17], [18]:

Create the decision matrix. Experts are created the decision matrix using the IVNNs between the criteria and alternatives.

Normalize the decision matrix

$$x_{ij}=\frac{y_{ij}}{\sum_{i=1}^{m}y_{ij}} \tag{9}$$

Where y_{ij} refers to the value in the decision matrix, $i=1,\dots,m; j=1,\dots,n$

Determine the entropy

$$e_j = -t \sum_{i=1}^{m} x_{ij} \ln x_{ij} \tag{10}$$

$$t = \frac{1}{\ln m} \tag{11}$$

Compute the criteria weights

$$w_j = \frac{1-e_j}{\sum_{j=1}^{n} 1-e_j} \tag{12}$$

IVN-CoCoSo Method

This section shows the steps of the CoCoSo method to rank the alternatives[19], [20].
Normalize the decision matrix by the CoCoSo method for positive and negative criteria.

$$k_{ij} = \frac{y_{ij} - \min_i y_{ij}}{\max_i y_{ij} - \min_i y_{ij}} \tag{13}$$

$$k_{ij} = \frac{\max_i y_{ij} - y_{ij}}{\max_i y_{ij} - \min_i y_{ij}} \tag{14}$$

Determine the power weight of decision matrix and total weighted decision matrix

$$U_i = \sum_{j=1}^{n} w_j k_{ij} \tag{15}$$

$$Q_i = \sum_{j=1}^{n} (k_{ij})^{w_j} \tag{16}$$

Compute the relative weights

$$H_{ia} = \frac{U_i + Q_i}{\sum_{i=1}^{m} (Q_i + U_i)} \tag{17}$$

$$H_{ib} = \frac{U_i}{\min_i U_i} + \frac{Q_i}{\min_i Q_i} \tag{18}$$

$$H_{ic} = \frac{ב(U_i) + (1-ב)(Q_i)}{\left(ב \max_i U_i + (1-ב) \max_i Q_i\right)} \quad 0 \le ב \le 1 \tag{19}$$

Rank the alternatives such as

$$H_i = (H_{ia} H_{ib} H_{ic})^{\frac{1}{3}} + \frac{1}{3}(H_{ia} + H_{ib} + H_{ic}) \tag{20}$$

The aim of this study is to utilize a deep-federated approach for identifying malicious software in mobile environments. This section gives an overview of CNN-Fed model shown in Figure 1, and discusses data preparation, modelling, optimization, and federated training processes:

Data Preparation: This stage entails changing of data, generating numerical data and applying standardization and normalization methods.

General Overview: At the server level, the proposed CNN-Fed model has a global model, which is a convolutional neural network and an aggregator. The server shares a pre-trained global model with all clients who train their local models with their own data. Clients then send their updated parameters back to the server, which combines all local models' outputs. The input of the model contains logs on mobile network traffic while its output is classified into zero for benign traffic or one for malicious traffic.

A Suggested Federated Learning Procedure: To begin the training process for the CNN-Fed model, global CNN model parameters are initially set on a global server. These are altered using an optimization algorithm thus starting the training phase. The process of training occurs in rounds that are connected and represented by n_rounds. Each round involves a random selection of clients who participate to ensure heterogeneous client representation called n_client. During this selection, clients are notified about the current global model's parameters, which would help, make their training uniform based on the same basic conditions. Through this, they pass their data through local training for n_epocs epochs and share it with the global server after updating its parameters; this is what happens during local training phases. Clients undertake a local training phase to adjust the models according to their data properties — optimizing their performance. Several rounds of communication, local trainings, and model aggregations later resulted in finalizing them. They reflect the combined efforts of all contributing clients to develop a common pool of knowledge and expertise in solving different computing problems globally. Such parameters can be used for many other downstream tasks.

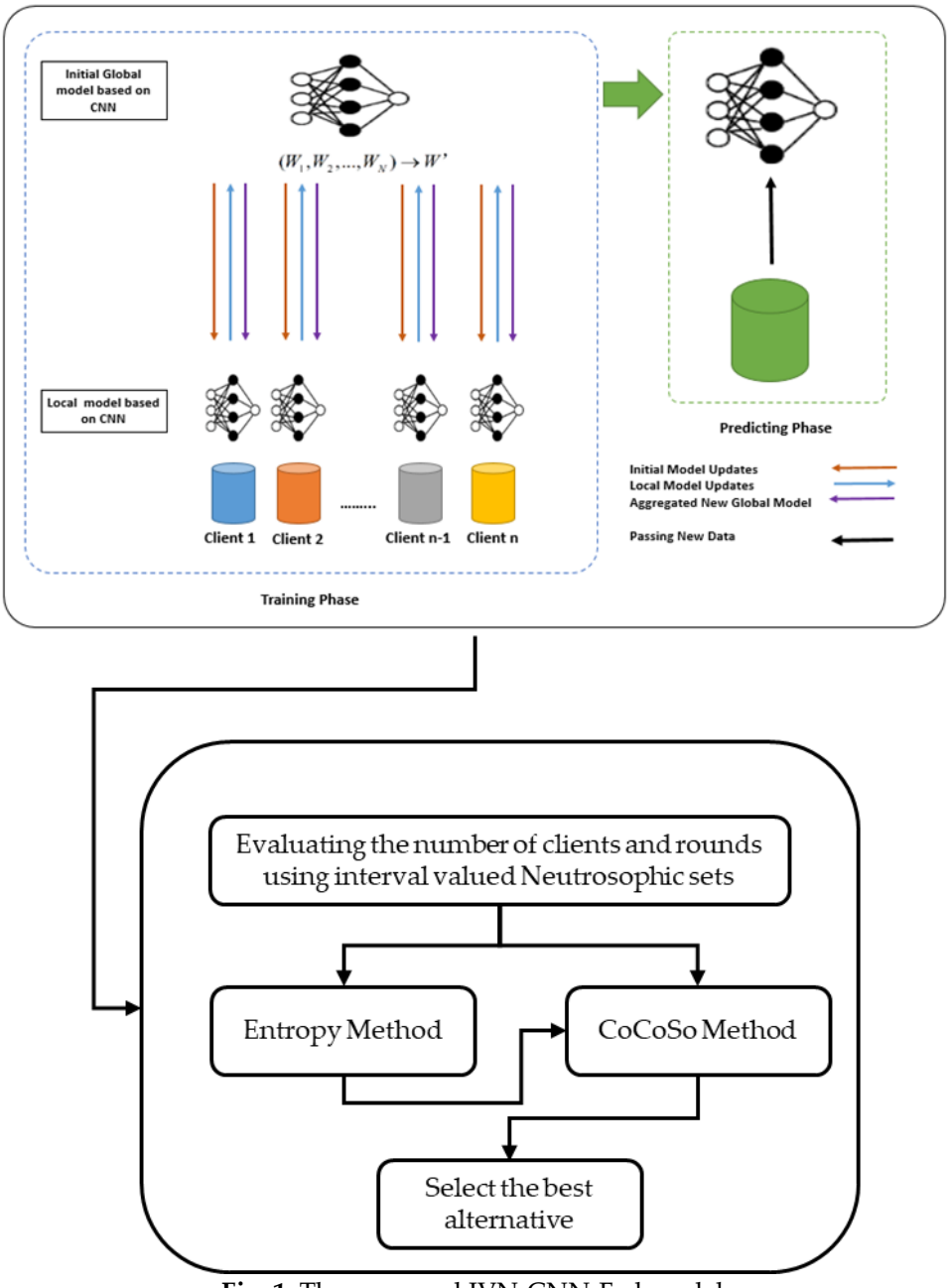

Fig 1. The proposed IVN-CNN-Fed model

. By enabling knowledge exchange among clients, the server eliminates the need for direct data access. As a result, there is no requirement to share data with other clients, and the server maintains data privacy.

3. Experiment

3.1 Dataset

Our work has used four publicly accessible datasets. Table 1 presents a summary of the datasets under consideration:

Table 1. Dataset description

Dataset Name	Ref	No. of Samples	No. of Features	Target Class	
				Benign	Malware
Drebin	[21]	15036	215	9476	5560
Malgenome	[22]	3799	215	2539	1260
Kronodrid	[23]	78137	463	36935	41382
Tunadromd	[24]	4465	241	903	3565

3.2 Performance Metric

The evaluation of the suggested CNN-Fed model for malware detection is assessed using different classification metrics. The primary metrics used in this study include:
True positive (γ): Occurrence in malware samples where the prediction was accurate.
False positive (μ): A test result indicates that a mobile device has malware while it does not contain malware.
True negative (σ): A true negative accurately predicts benign ware in samples.
False negative (ϱ): A test result that indicates that the mobile does not contain malware while the mobile does indeed contain malware.

• Precision:

Precision, also known as Positive Predictive Value, indicates the proportion of correct identifications. It is crucial to understand the reliability of the model in identifying malware.

$$\text{Precision} = \frac{\gamma}{(\gamma+\mu)} \quad (21)$$

Where γ represents true positives and μ represents false positives.

• Recall:

Recall, also known as Sensitivity or True Positive Rate, measures the ability of the model to correctly identify all relevant instances of the positive class (malware).

$$\text{Recall} = \frac{\gamma}{(\gamma+\rho)} \quad (22)$$

Where ϱ represents false negatives. A higher recall indicates that the model is effective at identifying most of the malware.

• Accuracy:

Accuracy measures the overall correctness of the model by calculating the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances.

$$\text{Accuracy}=(\gamma+\sigma)/(\gamma+\rho+\sigma+\mu) \quad (23)$$

• F1-Score:

The F1-Score is the harmonic mean of precision and recall, providing a single metric that balances both concerns. It is particularly useful when dealing with imbalanced datasets.

$$\text{F1-Score} = \frac{2*(\text{Precision}*\text{Recall})}{(\text{Precision}+\text{Recall})} \quad (24)$$

• False Positive Rate (FPR):

The False Positive Rate measures the proportion of benign instances that are incorrectly classified as malware. It is calculated as:

$$\text{FPR} = \frac{2*(\text{Precision}*\text{Recall})}{(\text{Precision}+\text{Recall})} \quad (25)$$

Where μ represents false positives and σ represents true negatives. A lower FPR indicates a better performance in distinguishing benign traffic from malware.

These metrics collectively offer a robust framework for evaluating the CNN-Fed model. By analysing these performance indicators, we can determine the efficacy of the proposed federated learning approach in accurately detecting malware in Mobile environments.

3.3 Results, Analysis, and Discussion

The experiments are conducted on four datasets for Android malware classification: Drebin, Malgenome, Kronodroid, and Tuandromd. Furthermore, a comparative analysis between the performance of our CNN-Fed model and other similar models was performed. The evaluation considered different numbers of clients and communication rounds to analyze the scalability and robustness of the models.

We examined three different scenarios based on client numbers, with 5, 10, and 15 clients. For each scenario, we conducted experiments with 10 and 20 rounds to assess the impact of the number of clients and rounds on model performance. Investigate how changes in client numbers and communication rounds influence the effectiveness of our proposed CNN-Fed model compared to Dw-FedAvg.

Figure 2 and Figure 3 show that CNN-Fed demonstrated competitive accuracy; on the Malgenome dataset (20 clients, five rounds), its highest accuracy was 0.9923. In general, DW-FedAvg exhibited larger loss values than CNN-Fed, suggesting that CNN-Fed performed better as a model in terms of minimising error. With twenty clients and five rounds in the Kronodroid dataset, CNN-Fed's lowest loss was 0.5151. With twenty clients and five rounds in the Tuandromd dataset, the lowest loss for DW-FedAvg was 0.3913. High F1 scores for both models indicate a favourable trade-off between precision and recall. With twenty clients and five rounds in the Tuandromd dataset, the greatest F1-score for DW-FedAvg was 0.9921. With twenty clients and five rounds in the Tuandromd dataset, the greatest F1-score for CNN-Fed was 0.9797. All scenarios exhibited great precision and recall, with both measures closely matched, suggesting consistent performance in minimising false negatives and recognizing real positives. The greatest accuracy and recall values for DW-FedAvg on the Malgenome dataset (10 clients, five rounds) were 0.9950 and 0.9950, respectively. CNN-Fed's best precision on the Malgenome dataset with ten clients and five rounds was 0.9934, while its greatest recall on the Tuandromd dataset (10 clients, five rounds) was 0.9950. With AUC values continuously over 0.99, the two models showed exceptional performance and a high degree of ability to distinguish between positive and negative classifications. With twenty clients and fifteen rounds in the Malgenome dataset showed that CNN-Fed's greatest AUC was 0.9997. FPR was low across all scenarios, with values mostly below 0.05, indicating that both models had a low rate of incorrectly classifying benign samples as malicious. The lowest FPR for CNN-Fed was 0.0054 on the Malgenome dataset (20 clients, five rounds).

The comparison in Figure 4 shows the Impact of the number of clients and number of rounds through the experiment.

Impact of the number of clients with every model CNN-Fed: Drebin Dataset: Accuracy slightly decreases as the number of clients increases (from 0.9845 with five clients to 0.9770 with fifteen clients in ten rounds). Malgenome Dataset: Accuracy decreases with more clients (0.9907 with five clients to 0.9892 with fifteen clients in ten rounds). Kronodroid Dataset: Accuracy decreases with more clients (0.9684 with five clients to 0.9628 with fifteen clients in ten rounds). Tuandromd Dataset: Accuracy decreases with more clients (0.9837 with five clients to 0.9778 with fifteen clients in ten rounds). DW-FedAvg: Drebin Dataset: Accuracy slightly decreases as the number of clients increases (from 0.9828 with five clients to 0.9734 with fifteen clients in ten rounds). Malgenome Dataset: Accuracy also shows a decrease with more clients (0.9943 with five clients to 0.9875 with fifteen clients in ten rounds). Kronodroid Dataset: Accuracy decreases with more clients (0.9597 with five clients to 0.9479 with fifteen clients in ten rounds). Tuandromd Dataset: Accuracy decreases with more clients (0.9861 with five clients to 0.9781 with fifteen clients in ten rounds).

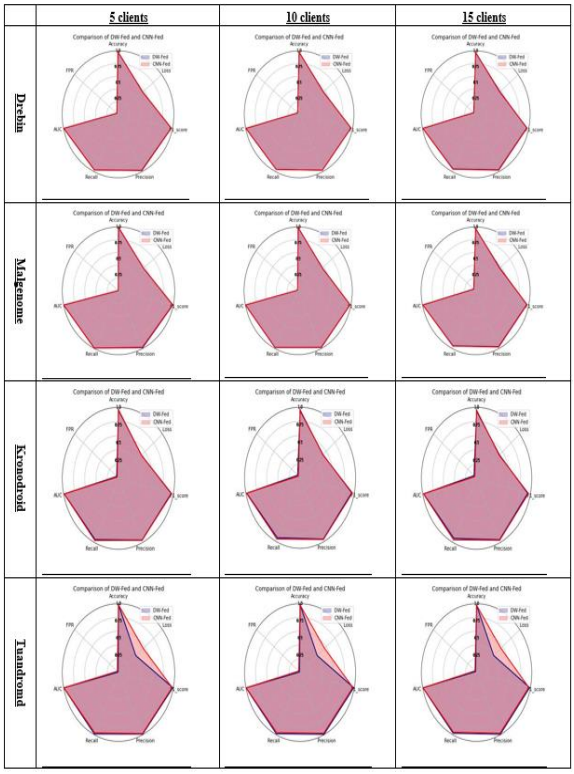

Figure 2: Performance comparison between CNN-Fed and DW-Fed for 10 Round

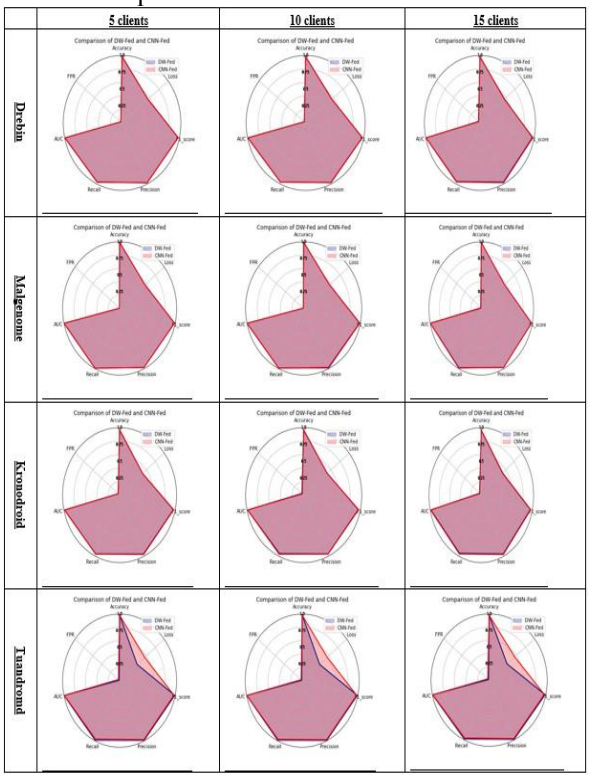

Figure 3: Performance comparison between CNN-Fed and DW-Fed for 20 Round

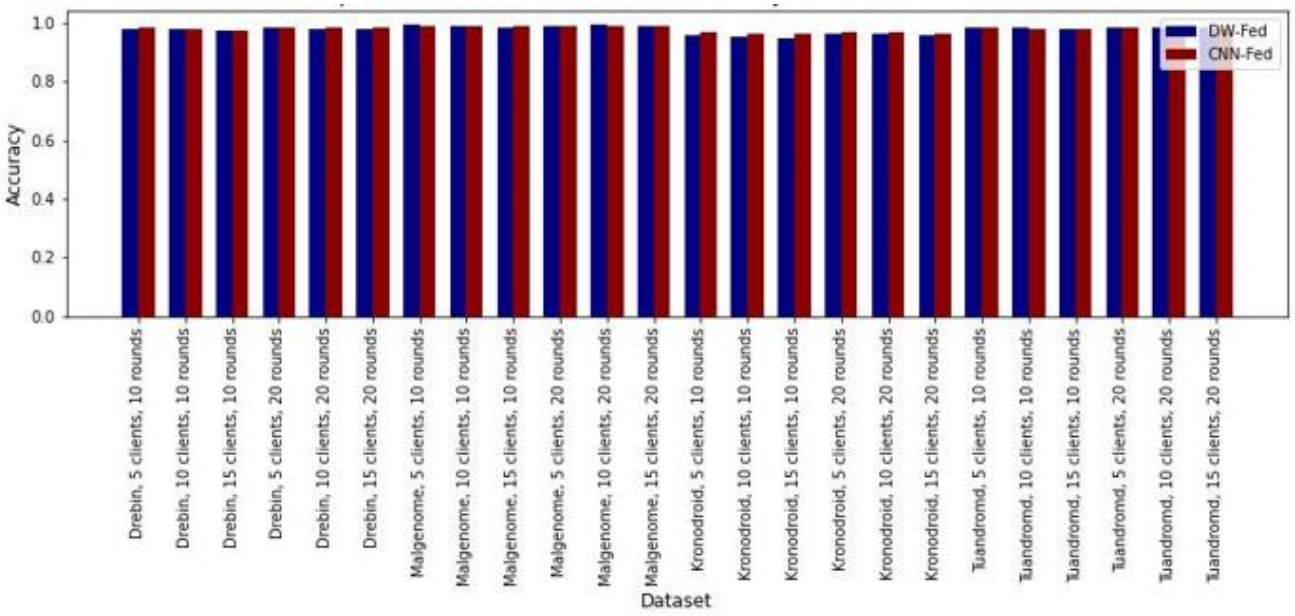

Figure 4: Accuracy comparison between CNN-Fed and DW-Fed

Impact of the number of rounds with every model CNN-Fed: Drebin Dataset: Accuracy increases with more rounds (from 0.9845 in 10 rounds to 0.9850 in 20 rounds with five clients). Malgenome Dataset: Accuracy increases with more rounds (from 0.9907 in 10 rounds to 0.9923 in 20 rounds with five clients). Kronodroid Dataset: Accuracy increases with more rounds (from 0.9684 in 10 rounds to 0.9713 in 20 rounds with five clients). DW-FedAvg: Drebin Dataset: Accuracy increases with more rounds (from 0.9828 in 10 rounds to 0.9841 in 20 rounds with five clients). Malgenome Dataset: Accuracy increases with more rounds (from 0.9943 in 10 rounds to 0.9923 in 20 rounds with five clients). Kronodroid Dataset: Accuracy increases with more rounds (from 0.9597 in 10 rounds to 0.9662 in 20 rounds with five clients). Tuandromd Dataset: Accuracy increases with more rounds (from 0.9861 in 10 rounds to 0.9871 in 20 rounds with five clients).

Then we evaluate the number of clients and rounds using the IVN-Entropy-CoCoSo method. The Entropy method is used to compute the criteria weights. Then we rank the alternatives using the CoCoSo method. The criteria of this study are accurate, precision, recall, f1-score, and false positive rate. The alternatives are shown the number of client and number of rounds.

In the first dataset

Results of IVN-Entropy

We show the results of the Entropy method. We applied the Entropy and CoCoSo methods into four datasets to show the criteria weights and ranking the alternatives.

Three experts are created the decision matrix using the IVNNs as shown in Table 2. Then we apply the score function to obtain the crisp values. Then we combine the decision matrix into a single matrix.

Eq. (9) is used to normalize the decision matrix as shown in Table 3.

Then we determine the entropy using Eqs. (10 and 11).

Then we compute the criteria weights using Eq. (12).

Table 2. Decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.6],[0.5,0.6],[0.4,0.5])
$IVNA_2$	([0.6,0.7],[0.4,0.5],[0.3,0.4])	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])
$IVNA_3$	([0.5,0.6],[0.5,0.6],[0.4,0.5])	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.6],[0.5,0.6],[0.4,0.5])

IVNA4	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.1,0.2],[0.1,0.2],[0.8,0.9])
IVNA5	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.6],[0.5,0.6],[0.4,0.5])	([0.6,0.7],[0.4,0.5],[0.3,0.4])
IVNA6	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.6,0.7],[0.4,0.5],[0.3,0.4])
	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.6],[0.5,0.6],[0.4,0.5])
IVNA2	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.5,0.6],[0.5,0.6],[0.4,0.5])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])
IVNA3	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.5,0.6],[0.5,0.6],[0.4,0.5])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.6],[0.5,0.6],[0.4,0.5])
IVNA4	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.6,0.7],[0.4,0.5],[0.3,0.4])	([0.1,0.2],[0.1,0.2],[0.8,0.9])
IVNA5	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.6],[0.5,0.6],[0.4,0.5])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.2,0.3],[0.3,0.4],[0.7,0.8])
IVNA6	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.6,0.7],[0.4,0.5],[0.3,0.4])	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])
	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	([0.6,0.7],[0.4,0.5],[0.3,0.4])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.5],[0.6,0.7],[0.4,0.5])
IVNA2	([0.6,0.7],[0.4,0.5],[0.3,0.4])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])
IVNA3	([0.1,0.2],[0.1,0.2],[0.8,0.9])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.3,0.4],[0.4,0.5],[0.6,0.7])
IVNA4	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.2,0.3],[0.3,0.4],[0.7,0.8])	([0.2,0.3],[0.3,0.4],[0.7,0.8])
IVNA5	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.6,0.7],[0.4,0.5],[0.3,0.4])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.5,0.5],[0.6,0.7],[0.4,0.5])
IVNA6	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.5,0.5],[0.6,0.7],[0.4,0.5])	([0.3,0.4],[0.4,0.5],[0.6,0.7])	([0.4,0.5],[0.5,0.6],[0.5,0.6])	([0.4,0.5],[0.5,0.6],[0.5,0.6])

Table 3. Normalized decision matrix.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0.150402	0.153207	0.183711	0.185383	0.192135
IVNA2	0.205511	0.160333	0.132272	0.153298	0.168539
IVNA3	0.144661	0.157957	0.183711	0.185383	0.182022
IVNA4	0.176808	0.13658	0.169014	0.170529	0.093258
IVNA5	0.174512	0.183492	0.18861	0.160428	0.179213
IVNA6	0.148106	0.208432	0.142682	0.144979	0.184831

Results of IVN-CoCoSo Method

We state with the combined decision matrix. Then we normalize the decision matrix using Eqs. (13 and 14) as shown in Table 4.

Then we determine the power weight of decision matrix and total weighted decision matrix suing Eq. (15 and 16) as shown in Tables 5 and 6.

Then we compute the relative weights using Eqs. (17-19)

Then we rank the alternatives based on Eq. (20).

Table 4. Normalized decision matrix by CoCoSo.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0	0.326087	0.875	1	0.935065
IVNA2	1	0	0	0.4375	0.688312
IVNA3	0.796407	0	0.875	1	0.935065
IVNA4	0.700599	0.021739	0.4375	0.739583	0
IVNA5	0.676647	0	1	0.71875	0.939394
IVNA6	0.401198	1	0.322917	0	1

Table 5. weighed normalized decision matrix.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0	0.057468	0.142019	0.07846	0.404042
IVNA2	0.150895	0	0	0.034326	0.297419
IVNA3	0.120174	0	0.142019	0.07846	0.404042

$IVNA_4$	0.105717	0.003831	0.07101	0.058028	0
$IVNA_5$	0.102103	0	0.162308	0.056393	0.405912
$IVNA_6$	0.060539	0.176237	0.052412	0	0.4321

Table 6. Power weighted normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.000	0.820789	0.97856	1	0.971406
$IVNA_2$	1	0	0	0.937197	0.850956
$IVNA_3$	0.966233	0	0.97856	1	0.971406
$IVNA_4$	0.947724	0.509286	0.874436	0.976609	0
$IVNA_5$	0.942763	0	1	0.974422	0.973347
$IVNA_6$	0.871262	1	0.83238	0	1

In the second dataset

Table 7 shows the normalization decision matrix.

Then we determine the entropy using Eqs. (10 and 11).

Then we compute the criteria weights using Eq. (12).

Table 7. Normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.175758	0.148276	0.185529	0.191882	0.192135
$IVNA_2$	0.202424	0.185632	0.150897	0.158672	0.168539
$IVNA_3$	0.117576	0.183333	0.197279	0.186962	0.182022
$IVNA_4$	0.198182	0.144828	0.166976	0.190652	0.093258
$IVNA_5$	0.149697	0.136207	0.155226	0.121771	0.179213
$IVNA_6$	0.156364	0.201724	0.144094	0.150062	0.184831

Then we normalize the decision matrix using Eqs. (13 and 14) as shown in Table 8 by the CoCoSo method.
Then we determine the power weight of decision matrix and total weighted decision matrix suing Eq. (15 and 16) as shown in Tables 9 and 10.

Then we compute the relative weights using Eqs. (17-19)

Then we rank the alternatives based on Eq. (20).

Table 8. Normalized decision matrix by CoCoSo.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.398496	0.557143	0.842975	0.791667	0.935065
$IVNA_2$	0.87218	0.847619	0.380165	0.541667	0.688312
$IVNA_3$	0	0.938095	1	0.736111	0.935065
$IVNA_4$	1	0.342857	0.719008	1	0
$IVNA_5$	0.398496	0	0	0	0.939394
$IVNA_6$	0.481203	1	0.404959	0.347222	1

Table 9. weighed normalized decision matrix.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0.092222	0.087616	0.079111	0.142454	0.315609
IVNA2	0.201844	0.133296	0.035678	0.097468	0.232323
IVNA3	0	0.147525	0.093848	0.132457	0.315609
IVNA4	0.231425	0.053918	0.067477	0.179942	0
IVNA5	0.092222	0	0	0	0.31707
IVNA6	0.111362	0.15726	0.038004	0.06248	0.337526

Table 10. Power weighted normalized decision matrix.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0.808	0.912117	0.984097	0.958834	0.977594
IVNA2	0.968846	0.974336	0.913233	0.895545	0.881553
IVNA3	0	0.990001	1	0.946363	0.977594
IVNA4	1	0.845069	0.969516	1	0
IVNA5	0.808217	0	0	0	0.979119
IVNA6	0.844273	1	0.918663	0.826677	1

In the third dataset

Table 11 shows the normalization decision matrix.

Then we determine the entropy using Eqs. (10 and 11).

Then we compute the criteria weights using Eq. (12).

Table 11. Normalized decision matrix.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0.154572	0.14527	0.183711	0.187952	0.165637
IVNA2	0.211209	0.152027	0.132272	0.146988	0.152723
IVNA3	0.129204	0.163851	0.183711	0.166265	0.152723
IVNA4	0.171091	0.14527	0.169014	0.155422	0.147108
IVNA5	0.171091	0.195946	0.18861	0.177108	0.19708
IVNA6	0.162832	0.197635	0.142682	0.166265	0.184728

Then we normalize the decision matrix using Eqs. (13 and 14) as shown in Table 12 by the CoCoSo method. Then we determine the power weight of decision matrix and total weighted decision matrix suing Eq. (15 and 16) as shown in Tables 13 and 14.

Then we compute the relative weights using Eqs. (17-19)

Then we rank the alternatives based on Eq. (20).

Table 12. Normalized decision matrix by CoCoSo.

	IVNC1	IVNC2	IVNC3	IVNC4	IVNC5
IVNA1	0	0.176991	0.875	1	0.693333
IVNA2	1	0	0	0	0.186667
IVNA3	0.540323	0.469027	0.875	0.512195	0

$IVNA_4$	0.798387	0.176991	0.4375	0.512195	0.426667
$IVNA_5$	0.798387	0.628319	1	0.780488	1
$IVNA_6$	0.685484	1	0.322917	0.219512	1

Table 13. weighed normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0	0.041333	0.204779	0.086499	0.110918
$IVNA_2$	0.285961	0	0	0	0.029862
$IVNA_3$	0.154511	0.109531	0.204779	0.044304	0
$IVNA_4$	0.228308	0.041333	0.10239	0.044304	0.068257
$IVNA_5$	0.228308	0.146731	0.234033	0.067511	0.159977
$IVNA_6$	0.196022	0.233529	0.075573	0.018988	0.159977

Table 14. Power weighted normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.000	0.667382	0.969232	1	0.943093
$IVNA_2$	1	0	0	0	0.764517
$IVNA_3$	0.838589	0.837944	0.969232	0.943771	0
$IVNA_4$	0.937642	0.667382	0.824094	0.943771	0.872615
$IVNA_5$	0.937642	0.897158	1	0.978791	1
$IVNA_6$	0.897639	1	0.767557	0.877076	1

In the fourth dataset

Table 15 shows the normalization decision matrix.

Then we determine the entropy using Eqs. (10 and 11).

Then we compute the criteria weights using Eq. (12).

Table 15. Normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.180193	0.175749	0.225225	0.219873	0.239161
$IVNA_2$	0.229023	0.134877	0.162162	0.154334	0.20979
$IVNA_3$	0.114168	0.132153	0.145646	0.179704	0.125874
$IVNA_4$	0.136176	0.156676	0.148649	0.134602	0.116084
$IVNA_5$	0.189821	0.161444	0.183183	0.139535	0.173427
$IVNA_6$	0.150619	0.239101	0.135135	0.171952	0.135664

Then we normalize the decision matrix using Eqs. (13 and 14) as shown in Table 16 by the CoCoSo method.
Then we determine the power weight of decision matrix and total weighted decision matrix suing Eq. (15 and 16) as shown in Tables 17 and 18.

Then we compute the relative weights using Eqs. (17-19)

Then we rank the alternatives based on Eq. (20).

Table 16. Normalized decision matrix by CoCoSo.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$

$IVNA_1$	0.260417	0.557143	1	1	1
$IVNA_2$	1	0	0.314465	0.363014	0.736111
$IVNA_3$	0.130208	0	0	0.219178	0
$IVNA_4$	0	0.357143	0.157233	0	0
$IVNA_5$	0.557292	0	0.314465	0	0.347222
$IVNA_6$	0.203125	1	0.333333	0.171233	0

Table 17. weighed normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.058674	0.105871	0.138234	0.126807	0.319626
$IVNA_2$	0.225308	0	0.04347	0.046033	0.23528
$IVNA_3$	0.029337	0	0	0.027793	0
$IVNA_4$	0	0.067866	0.021735	0	0
$IVNA_5$	0.125562	0	0.04347	0	0.110981
$IVNA_6$	0.045766	0.190025	0.046078	0.021714	0

Table 18. Power weighted normalized decision matrix.

	$IVNC_1$	$IVNC_2$	$IVNC_3$	$IVNC_4$	$IVNC_5$
$IVNA_1$	0.738	0.894803	1	1	1
$IVNA_2$	1	0	0.852212	0.879418	0.906717
$IVNA_3$	0.631715	0	0	0.824914	0
$IVNA_4$	0	0.822297	0.774346	0	0
$IVNA_5$	0.876578	0	0.852212	0	0.713126
$IVNA_6$	0.698286	1	0.859104	0.799491	0

Then we compute the criteria in each dataset as shown in Fig 5. Then we ranked the alternatives as shown in Fig 6. We show the large number of clients and large number of rounds are the best alternative in each dataset.

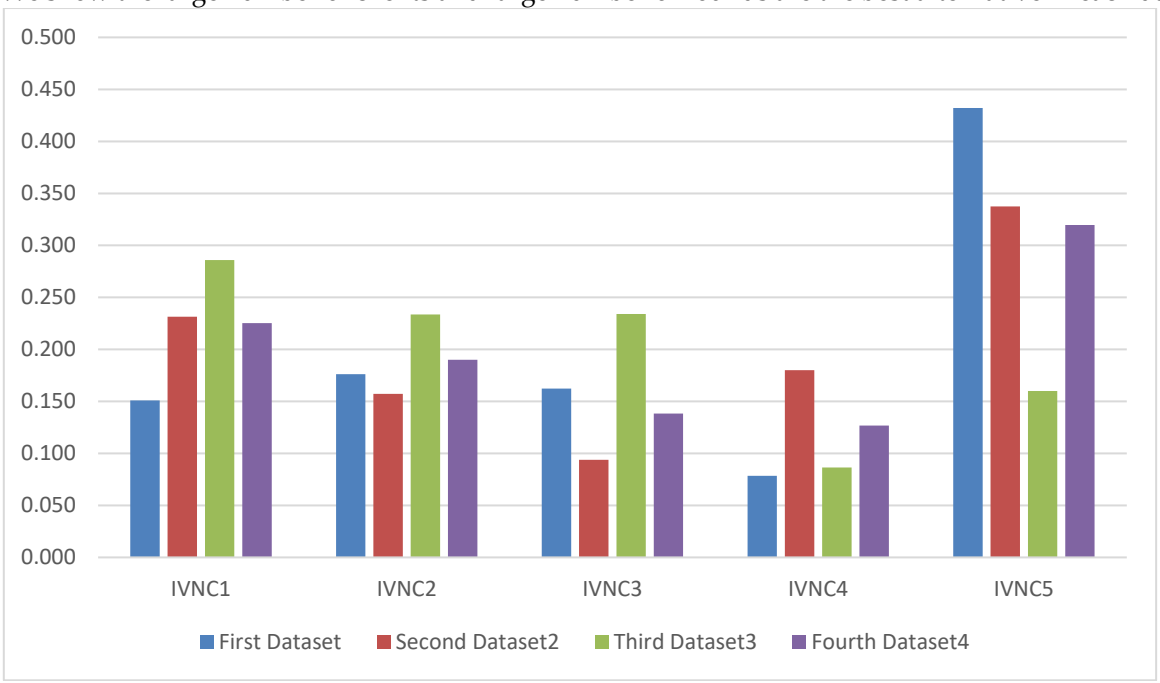

Fig 5. Criteria weights of four datasets.

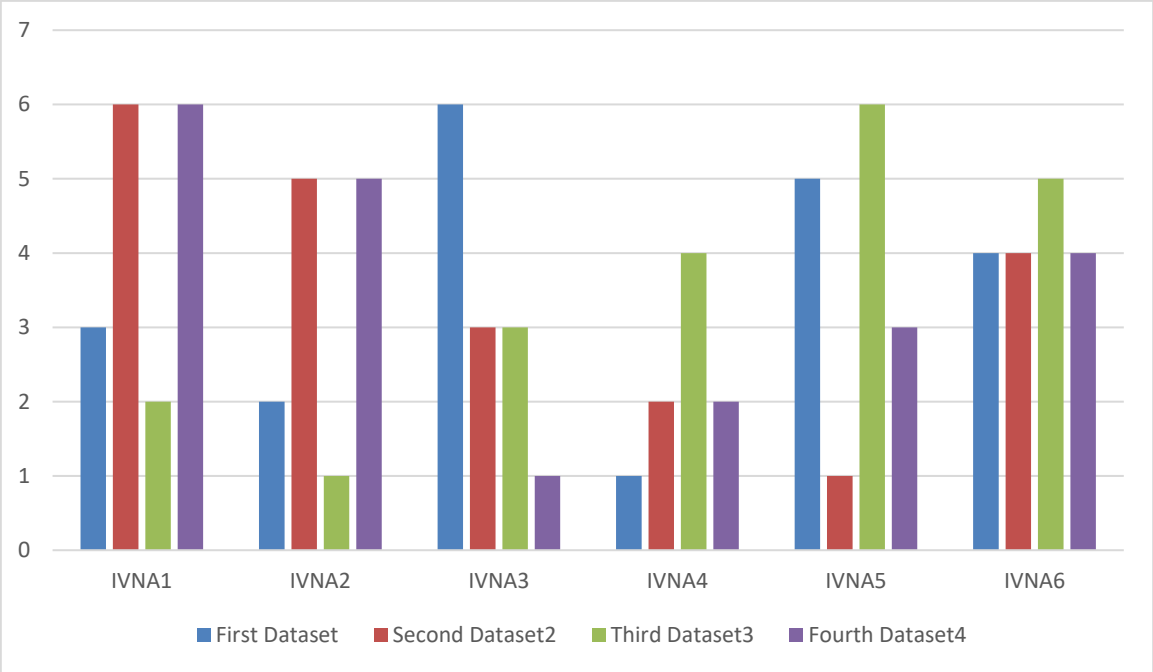

Fig 6. Rank the alternatives of four datasets.

4. Conclusions

In this study, we introduced CNN-Fed, a novel federated learning framework based on CNN, for mobile malware detection. The primary objective was to improve the accuracy of the global classifier while preserving data privacy through decentralized model training. By leveraging a dynamic weighted aggregation mechanism, CNN-Fed improves the contribution of high-performing local models to the global model, resulting in superior classification performance. Using four datasets for Android malware classification: Malgenome, Drebin, Kronodroid, and Tuandromd We evaluated CNN-Fed and compared the results with the Dw-FedAvg technique across all datasets. These findings prove that CNN-Fed is a promising approach for enhancing the accuracy of federated learning models in malware detection applications, making it a valuable tool for cybersecurity. We used the interval valued neutrosophic set (IVNSs) methodology to evaluate the number of clients and number of rounds. We used five criteria such as accuracy, precision, recall, f1-score, and false positive rate. Then we used two MCDM methods such as Entropy to compute the criteria weights and the CoCoSo method to rank the alternatives. IVNSs are uses to deal with indeterminacy. We applied these methods into four datasets. The results show the larger clients and rounds are the best alternatives.

References

[1] R. Kumar, P. Kumar, R. Tripathi, G. P. Gupta, S. Garg, and M. M. Hassan, "A distributed intrusion detection system to detect DDoS attacks in blockchain-enabled IoT network," *J. Parallel Distrib. Comput.*, vol. 164, pp. 55–68, 2022.

[2] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *J. Syst. Archit.*, vol. 112, p. 101861, 2021.

[3] S. K. Smmarwar, G. P. Gupta, and S. Kumar, "Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review," *Telemat. Informatics Reports*, p. 100130, 2024.

[4] A. H. Alharbi, H. M. Jabali, P. Rajan, and A. A. Salamai, "Evaluation Challenges of Leadership

Management in the Energy Sector using Multi-Criteria Decision Making Approach," *Multicriteria Algorithms with Appl.*, vol. 4, pp. 16–27, 2024.

[5] M. Mohamed and A. Elsayed, "A novel multi-criteria decision making approach based on bipolar neutrosophic set for evaluating financial markets in egypt," *Multicriteria Algorithms with Appl.*, vol. 5, pp. 1–17, 2024.

[6] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, 1965.

[7] L. A. Zadeh, "Fuzzy sets as a basis for a theory of possibility," *Fuzzy sets Syst.*, vol. 1, no. 1, pp. 3–28, 1978.

[8] F. Smarandache, *A unifying field in logics: neutrosophic logic. Neutrosophy, neutrosophic set, neutrosophic probability: neutrsophic logic. Neutrosophy, neutrosophic set, neutrosophic probability*. Infinite Study, 2005.

[9] F. Smarandache, *Introduction to neutrosophic measure, neutrosophic integral, and neutrosophic probability*. Infinite Study, 2013.

[10] W. B. V. Kandasamy and F. Smarandache, *Fuzzy Interval Matrices, Neutrosophic Interval Matrices and Their Applications*. Infinite Study, 2006.

[11] H. Wang, F. Smarandache, R. Sunderraman, and Y.-Q. Zhang, *interval neutrosophic sets and logic: theory and applications in computing: Theory and applications in computing*, vol. 5. Infinite Study, 2005.

[12] S. Broumi, I. Deli, and F. Smarandache, "N-valued interval neutrosophic sets and their application in medical diagnosis," *Collect. Pap. Vol. XIV Neutrosophics other Top.*, pp. 45–69, 2022.

[13] F. Deldar and M. Abadi, "Deep learning for zero-day malware detection and classification: A survey," *ACM Comput. Surv.*, vol. 56, no. 2, pp. 1–37, 2023.

[14] I. Tareq, B. M. Elbagoury, S. El-Regaily, and E.-S. M. El-Horbaty, "A survey about deep learning and federated Learning in cyberse-curity," *Period. Eng. Nat. Sci.*, vol. 12, no. 1, pp. 75–100, 2024.

[15] I. Deli, "Interval-valued neutrosophic soft sets and its decision making," *Int. J. Mach. Learn. Cybern.*, vol. 8, pp. 665–676, 2017.

[16] H. Zhang, J. Wang, and X. Chen, "An outranking approach for multi-criteria decision-making problems with interval-valued neutrosophic sets," *Neural Comput. Appl.*, vol. 27, pp. 615–627, 2016.

[17] P. P. Dwivedi and D. K. Sharma, "Application of Shannon entropy and CoCoSo methods in selection of the most appropriate engineering sustainability components," *Clean. Mater.*, vol. 5, p. 100118, 2022.

[18] P. P. Dwivedi and D. K. Sharma, "Application of Shannon Entropy and COCOSO techniques to analyze performance of sustainable development goals: The case of the Indian Union Territories," *Results Eng.*, vol. 14, p. 100416, 2022.

[19] Y. Zhu, S. Zeng, Z. Lin, and K. Ullah, "Comprehensive evaluation and spatial-temporal differences analysis of China's inter-provincial doing business environment based on Entropy-CoCoSo method," *Front. Environ. Sci.*, vol. 10, p. 1088064, 2023.

[20] D. Stanujkic, G. Popovic, E. K. Zavadskas, D. Karabasevic, and A. Binkyte-Veliene, "Assessment of progress towards achieving Sustainable Development Goals of the 'Agenda 2030' by using the CoCoSo

and the Shannon Entropy methods: The case of the EU Countries," *Sustainability*, vol. 12, no. 14, p. 5717, 2020.

[21] Y. Zhang, C. Jiang, B. Yue, J. Wan, and M. Guizani, "Information fusion for edge intelligence: A survey," *Inf. Fusion*, vol. 81, pp. 171–186, 2022.

[22] S. Y. Yerima and S. Sezer, "Droidfusion: A novel multilevel classifier fusion approach for android malware detection," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 453–466, 2018.

[23] A. Guerra-Manzanares, H. Bahsi, and S. Nõmm, "Kronodroid: Time-based hybrid-featured dataset for effective android malware detection and characterization," *Comput. Secur.*, vol. 110, p. 102399, 2021.

[24] P. Borah, D. K. Bhattacharyya, and J. K. Kalita, "Malware dataset generation and evaluation," in *2020 IEEE 4th Conference on Information & Communication Technology (CICT)*, IEEE, 2020, pp. 1–6.

Received: Oct 18, 2024. Accepted: March 21, 2025