Power System Protection and Control
ISSN:1674-3415

A SHARPNESS-AWARE FEDERATED LEARNING FRAMEWORK WITH MC-DROPOUT FOR CNN-BASED MALWARE CLASSIFICATION TO MITIGATE OVERFITTING AND MODEL UNCERTAINTY IN HETEROGENEOUS ENVIRONMENTS

Sadeq Thamer Hlama1, Abolfazl Gandomi2*, Zuhair Hussein Ali3, and Mohammadreza Soltanaghaei4

[1]PhD Candidate, Department of Computer Engineering, Institute of Artificial Intelligence and Social and Advanced Technologies, Isf.C., Islamic Azad University, Isfahan, Iran.
[2]*Department of Computer Engineering, Ya.C., Islamic Azad University, YAZD, Iran.
Abolfazl.Gandomi@iau.ac.ir, https://orcid.org/0000-0003-3353-3762
(**Correspond Author**)
[3]Department of Computer Science, College of Education, Mustansiriyah University, Iraq.
Zuhair72h@uomustansiriyah.edu.iq
[4]Department of Computer Engineering, Institute of Artificial Intelligence and Social and Advanced Technologies, Isf.C., Islamic Azad University, Isfahan, Iran.
soltan@iau.ac.ir

Received:27 September 2025 **Revised**:29 October 2025 **Accepted**: 10 December 2025

***ABSTRACT*:**

Machine learning techniques are not yet widely applicable in the field of malware detection, especially because of data heterogeneity, privacy issues, and the overfitting problem of the model. The presented paper introduces an innovative federated learning framework that combines CNN-based models with Sharpness Aware Minimization (SAM) and Monte Carlo Dropout (MC-Dropout) that overcome these issues. Particularly, the framework is effective operating in decentralized and non-IID settings since it encourages flat minima in the loss landscape, thus boosting model generalization and robustness. Also, predictive uncertainty can be measured due to the utilization of MC-Dropout, a crucial feature when it comes to enhancing the reliability of a decision in the field of cybersecurity. Experimental findings confirm that the suggested method remains considerably superior to the traditional techniques, having high levels of accuracy (up to 99.86%) and precision (99.87%) along with the meaningful estimation of uncertainties. The prospect of unifying the concepts of sharpness-aware optimization and uncertainty quantification under a federated architecture presents a potential solution towards having reliable and privacy-preserved malware detection in heterogenous operational environments.

INTRODUCTION

The wide-spread development of digital infrastructure has also caused a similar growth of cyber threats, in particular the occurrence of malware. Malware, Malicious software is a heterogeneous category of programs or scripts that include viruses, worms, and trojans, as well as spyware that enter, harm, or otherwise paralyze a computer system without the owner's authorization [1-3]. Given that the sophistication and obfuscation of the modern malware is increasing, finding proper and scalable detection mechanisms has become an urgent need in cybersecurity research. The existing detection methods, especially signature-based systems, prove ineffective in the identification of novel or obfuscated threats, and thus a paradigm towards the usage of machine learning (ML)-based detection frameworks has been identified [4]. Specifically, Convolutional Neural Networks (CNNs) have become an effective solution to malware classification especially when executable binaries are converted into image-like forms [5].

Although CNNs have shown to be very accurate in detecting complicated patterns in malware image data, their practicality in mass and privacy-sensitive environment is limited. Training paradigms that involve centralization usually involve the consolidation of large volumes of data in a central server which poses significant threats in

DOI: 10.46121/pspc.53.4.20

terms of data privacy, ownership and regulation [6,7]. To address such issues, a new approach to decentralized training, Federated Learning (FL) has been proposed. FL helps inferentially robotize the training of local models at the edge devices or distributed nodes without moving the raw data, thus providing a potential way to privacy-preserving machine learning in delicate fields like healthcare, finance and cybersecurity [8].

Federated Learning has various benefits: it does not expose the users to the privacy invasion due to the retention of the raw data on the local level and minimizes the risks of the centralized data storage manner, as well as makes it easier to collaborate between dispersed stakeholders without providing them with sensitive data [9]. Nevertheless, FL has also some challenges such as statistical heterogeneity of data between different clients (non-IID data), overhead in communication, and convergence of models. Such issues may cause a slowdown in the performance of global models, particularly in such tricky domains as malware detection, where the data distribution can differ significantly between the organizations and devices [10,11].

Model uncertainty is another important but poorly addressed concern in federated malware identification. In security applications, simple production of a label is not enough, the confidence in the decision must be measured as well. In extreme situations, i.e., detecting zero-day malware or ransomware, the system needs to signal that it is correct in its decisions so that the correct response can be taken. Sadly, the vast majority of deep learning models in the traditional sense are deterministic and they lack the ability to estimate the uncertainty in their parameters which makes them hard to deploy in critical settings where reliability and explainability are paramount [12].

Addressing these gaps, this paper proposes a novel Sharpness-Aware Federated Learning framework designed to enhance malware classification performance under heterogeneous and privacy-sensitive conditions. Our approach incorporates Sharpness-Aware Minimization (SAM) at the client level to guide models toward flatter minima in the loss landscape, improving generalization and reducing sensitivity to local data perturbations caused by non-IID distributions. In parallel, the framework integrates Monte Carlo Dropout (MC-Dropout) during inference to quantify predictive uncertainty without incurring heavy computational costs or architectural complexity. Furthermore, to increase robustness in real-world deployment, the architecture supports asynchronous federated updates, mitigating the effects of client dropout and communication failures.

The major contributions of this study include:

1. Proposing a Sharpness-Aware Federated Learning Framework for Malware Detection under Data Heterogeneity:

The proposed framework leverages sharpness-aware optimization to guide local models toward flatter regions in the parameter space. This approach mitigates overfitting and client drift arising from non-IID data distributions in noisy and heterogeneous environments.

2. Uncertainty Modeling through Monte Carlo Dropout (MC-Dropout):

By integrating MC-Dropout, the framework enables estimation of predictive variance and entropy, thereby equipping the malware detection system with the ability to assess the confidence level of its decisions. This enhancement significantly improves the reliability and trustworthiness of the model.

3. Design and Implementation of a Fault-Tolerant Asynchronous Distributed Update Mechanism for Local Nodes:

The model training and update processes are carried out asynchronously and independently across local clients. This design ensures that the training process remains uninterrupted even in the event of local node failures, thereby enhancing the system's scalability and robustness in real-world distributed environments.

4. Simultaneous Integration of Sharpness-Aware Minimization (SAM) and MC-Dropout in a Unified Federated Framework:

For the first time, the proposed method combines sharpness-aware optimization (SAM) and uncertainty modeling via MC-Dropout within a unified federated learning system. This hybrid strategy improves both the accuracy and stability of the model in heterogeneous environments, while also enabling reliable uncertainty estimation—jointly boosting both the performance and dependability of the malware detection system.

RELATED WORKS

In [13] authors present SIM-FED, a novel framework that aims at detecting malware in the Internet of Things (IoT) contexts. In such a way, this solution is based on a combination of deep learning and federated learning to allow privacy-preserving analysis, which effectively prevents data sharing and increases security in a decentralized context. The model uses a lean 1D-CNN model with hyperparameters tuned, the slightest

DOI: 10.46121/pspc.53.4.20

preprocessing as well as a computational load. A number of federated aggregation algorithms are considered, FedAvg being finally selected to combine local model outputs. Evaluation shows that SIM-FED outperforms the current available models of deep and federated learning in all terms with a remarkable rate of accuracy at 99.52 percent.

In [14] authors consider the use of federated learning (FL) to build and share aggregated deep neural networks (DNNs) between independent, federated organizations. Their solution presupposes that every organization uses its own malware analysis platform in a Security Operations Center (SOC), and trains on them local DNNs on company-specific data. It is a design based on a cross-silo FL framework in which these locally trained models are integrated into a global DNN, further shared with all participants so that collaborative malware detection can be performed without sharing any raw data or features. This federated method is able to reach an accuracy that is as high as a non-federated and centralized DNN and goes over 93%.

In [15] authors study the feasibility of federated learning as a privacy-preserving alternative to classical centralized machine learning. Their work assesses the effectiveness of FL-based models against traditional non-federated methods that work on CIC-MalMem-2022. There were 22 models that were created, including both the use of feedforward neural networks and long short-term memory (LSTM) architectures, of which four models were trained non-federatedly. These results mean that federated learning provides high performance scores - an accuracy of 0.999 on binary-classification and 0.845 on multiclassification, even in cases where the distributions of users vary.

In [16] authors propose a malware detection model by using a simplified convolutional neural network (CNN) structure. Promoting the principle of minimalism, the proposed model is expected to have good classification results, despite the presence of imbalanced data. The main idea is to investigate the impact of different malware image sizes (32x32, 64x64, 128x128 and 256x256) have on the results of the detection. The findings indicate that reduced image resolutions, especially those of 3232 pixels, have better detection efficiency with accuracy of 99.601 percent. In addition, those which were developed using these smaller-sized images allowed fewer computational resources, which shows the appropriateness of this architecture in restricted resource scenarios.

In [17] authors suggest the convolutional neural network (CNN) called Image-based Malware Classification with Multi-scale Kernels (IMCMK) to improve the ability to detect malware variants by incorporating multi-scale convolutional kernels. The most significant part of the model is the Multi-scale Kernels (MK) block that consists of both large and small kernel convolutions and ShortCut connection to enhance the performance of classification. As a means to alleviate the computing overhead that arises through the large-sized kernels, the authors propose a Multi-scale Kernel Fusion (MKF) mechanism that minimizes overhead of parameters. By experimental results, it can be proved that IMCMK can perform high on malware family classification with an accuracy of 99.25 which is higher compared to other existing state-of-the-art methods.

Authors in [18] integrate federated learning and incremental learning to maintain privacy of users but enhance detection. Federated learning is employed to train a Multi-Layer Perceptron (MLP) model on decentralized devices without exchange of raw data, and stacking is an ensemble learning technique employed to allow incremental updates to the trained model. With CICMalDroid 2020 dataset, which are a set of static features, the proposed framework attains an accuracy of 96.49 percent.

In [19] authors introduce a new detection framework that exploits grey-scale image representation of malware along with an autoencoder-based deep learning model. The approach checks the possibility of applying reconstruction errors of the autoencoder to classify benign and malicious software, and harnessing dimensionality reduction aspect of the model to classify a task. The experiments implemented on a custom Android dataset prove that the model has an accuracy of 96% in detecting, as well as an F-score of nearly 96 %, which out-competes a number of traditional machine learning techniques.

Authors in [20] introduce a new vision-based methodology of IoT malware detection and classification in multiple classes with the help of deep transfer learning. The technique employs fine-tuning and ensembling techniques to improve performance without demolishing models and starting afresh. Particularly, it combines three well-trained CNN networks ResNet18, MobileNetV2, and DenseNet161 through a random forest voting process. The findings demonstrate that the model yielded 98.74 precision, 98.67 recall, 98.79 specificity, 98.70 f1-score, 98.65 Matthews correlation coefficient (MCC), 98.68 accuracy with a mean of 672 milliseconds per sample.

DOI: 10.46121/pspc.53.4.20

In [21] authors introduce hybrid deep learning architecture where VGG-16 and ResNet-50 will be combined to fight malware through improved classification. This combined model uses the merits of both architectures in order to enhance the accuracy of detection. It is tested on three reference datasets such as Microsoft BIG 2015, Malimg and MaleVis and the experimental results demonstrate its superior performance over the recent approaches 99.28% accuracy.

In [22] authors propose using a Convolutional Neural Networks (CNN) and Transformer-based detection mechanism to enhance efficiency and classification succinctness of malicious codes. This model presents a modification scheme which is a fusion module which redesigns the network to lower memory access expenses by eliminating residual connections. Also, during training by linear methods, this method uses overparametrization and deep kernel convolution training methods to increase precision. The authors preprocess data by using pixel-wise normalization of the size of the images and data augmentation to deal with loss of texture in the images due to scaling and the problem of class imbalance, which improves the representational power of the data features and prevents overfitting. As the experimental results have indicated, this hybrid model performs better than the recent state-of-the-art malicious code detection techniques in terms of accuracy and robustness.

PROPOSED METHOD

This work presents a novel federated learning framework for malware detection based on Convolutional Neural Networks (CNNs) in heterogeneous environments. The proposed framework, grounded in CNN-based federated learning, is specifically designed to mitigate the issues posed by decentralized and non-IID data, which are common in real-world malware detection scenarios. Traditional federated learning methods, such as FedAvg, often suffer from significant performance degradation when client data are noisy, structurally diverse, or statistically non-identical. These issues typically lead to local overfitting and client drift—where local models diverge significantly from the global model. Such inconsistency across clients hampers convergence and deteriorates the stability of the final global model.

To address these challenges, our framework incorporates a Sharpness-Aware Minimization (SAM) approach in local training. Instead of relying on standard optimization techniques, SAM directs model parameters toward flatter minima in the loss landscape, thereby reducing sensitivity to noise and improving generalization. In the proposed system, every client trains a lightweight CNN using its local dataset. During training, rather than applying straightforward gradient updates, the loss function is first evaluated under a small perturbation that increases sharpness, and the final gradients are then computed in this perturbed region. This strategy encourages the local model to avoid sharp minima and stabilize within flatter, more generalizable regions. Model data are transmitted to the central server for aggregation when local training is finished. Until the global model converges, this federated training procedure is repeated across a number of communication cycles. To further enhance model reliability, Monte Carlo Dropout (MC-Dropout) is employed in the final stage to quantify predictive uncertainty. Unlike conventional dropout, which is only active during training, MC-Dropout remains enabled during inference. This approach is particularly beneficial in heterogeneous or noisy environments, where data uncertainty is high. It helps reduce the risk of incorrect predictions, increases system reliability, and strengthens the model's resilience under uncertain or noisy conditions. A conceptual overview of the proposed sharpness-aware federated learning system with CNNs, involving k clients, a central server, and a global model G, is illustrated in Figure 1.

DOI: 10.46121/pspc.53.4.20

Power System Protection and Control
ISSN:1674-3415

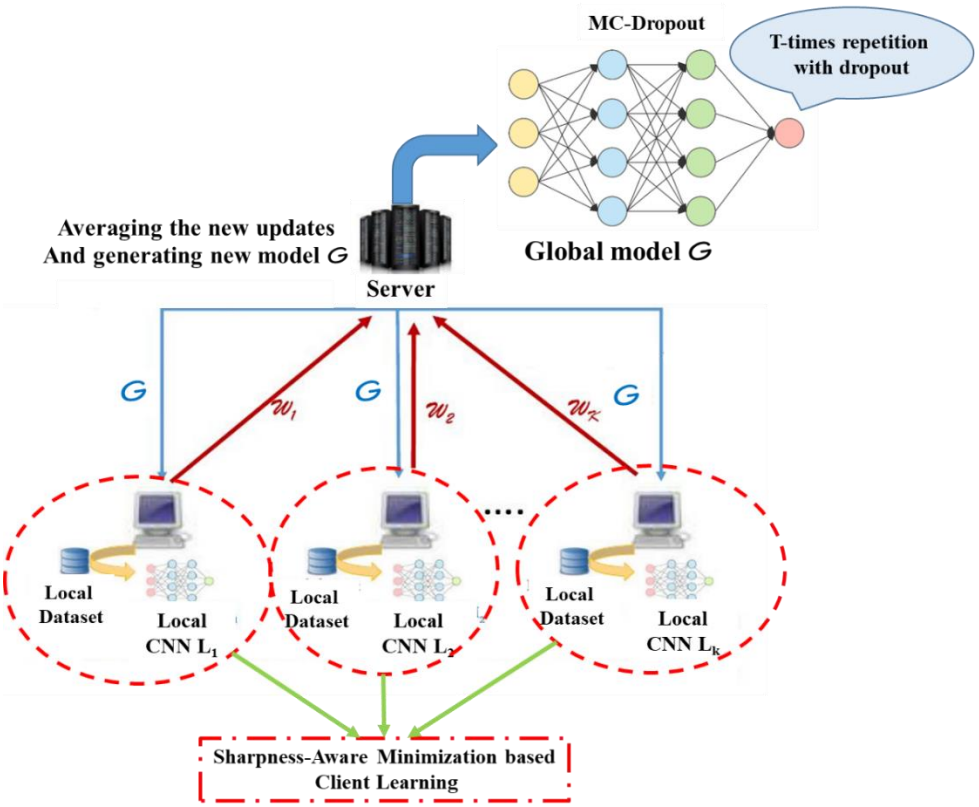

Figure 1. A conceptual overview of the proposed sharpness-aware federated learning system with CNNs

3.1. Base CNN Model Design

In this study, to detect and classify malware based on binary image representations from the MalImg dataset, a baseline Convolutional Neural Network (CNN) architecture has been designed. The network is structured to hierarchically extract both low-level and high-level features from the input images, thereby enabling accurate classification. The architecture of the proposed baseline CNN model is illustrated in Figure 2.

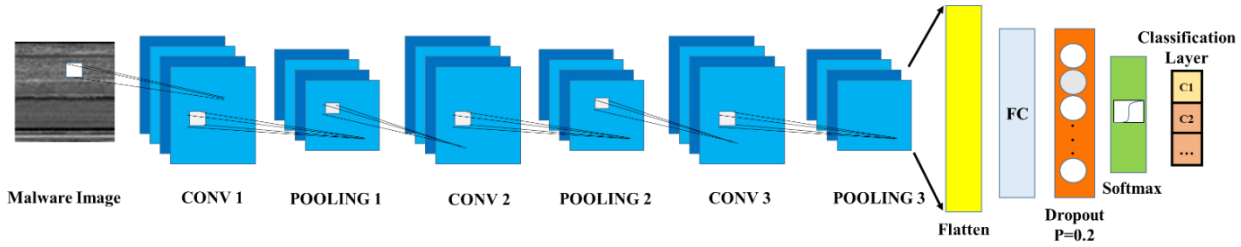

Figure 2. Architecture of the baseline CNN model

The model consists of three repeated blocks of Conv2D and MaxPooling2D layers.
The first block starts with a ReLU activation function after a convolutional layer with 32 (3×3) filters. Convolutional layers, through fixed-size filters (in this case, 3×3), extract local image features such as edges, corners, and texture patterns. Each convolutional operation, combined with a bias term, produces a new feature map from the input. The layer's operation can be mathematically expressed as:

$$O_k(i,j) = \sigma\left(\sum_{c=1}^{C}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} I_c\,(i+m.\,j+n)\cdot W_{k,c}(m.n) + b_k\right) \quad (1)$$

Where; I_c is the input image on channel c, $W_{k.c}$ is the filter corresponding to channel c and output channel k, $b_k$is the bias associated with filter k, and $\sigma(\cdot)$is the ReLU activation function defined as $\sigma(x) = \max(0.x)$. The output is then passed through a MaxPooling2D layer with a 2×2 window, which reduces the spatial dimensions (height and width) of the feature maps. This operation helps avoid overfitting and enhances the system's robustness to small spatial shifts. The pooling operation is defined as:

DOI: 10.46121/pspc.53.4.20

$$P(i,j) = \max_{\substack{0\le m<p \\ 0\le n<p}} O(i\cdot p + m.j\cdot p + n) \quad (2)$$

Where O is the input to the pooling layer, p is the pooling window size, and $P(i,j)$ is the maximum value in the corresponding block. In subsequent stages, the number of filters is increased to 64 and then 128, while the same convolution-pooling structure is maintained. This gradual increase in filters enhances the network's capacity to capture deeper and more abstract semantic features from the input images. The output of the final MaxPooling2D layer is passed to a Flatten layer, which converts the multi-dimensional feature map into a one-dimensional vector suitable for input to fully connected layers. Following this, a fully connected (Dense) layer with 128 neurons and a ReLU activation function is used to combine the extracted features and learn non-linear relationships among them. The output of each neuron in this layer is computed as:

$$h_j = \sigma\left(\sum_{i=1}^{n} w_{ji}\, x_i + b_j\right) \quad (3)$$

Here, x_i is the i-th input feature, w_{ji} is the weight connecting input neuron i to output neuron j, b_j is the bias term, and σ denotes the ReLU function. To mitigate overfitting and enhance generalization, a Dropout layer with a dropout rate of 0.5 is applied, randomly deactivating 50% of neurons during training. Its behavior can be expressed as:

$$\tilde{h}_i = r_i \cdot h_i \quad with \quad r_i \sim \text{Bernoulli}(1-p) \quad (4)$$

Where h_i is the original neuron output, h_i is the output after dropout, and r_i is a binary mask sampled from a Bernoulli distribution. Finally, the output layer consists of as many neurons as there are classes in the MalImg dataset and uses the Softmax activation function to convert the final outputs into a probability distribution over the classes.

$$\text{Softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}}, \quad i = 1, \ldots, C \quad (5)$$

Where z_i is the input to neuron i in the output layer, and C is the total number of malware classes.

3.2. Federated learning model based on CNN

In this work, a federated CNN architecture is employed for the detection and classification of different classes of malware. In the suggested model, many local CNNs (worker) perform malware classification and update their parameters via a central server located at the master node. This decentralized structure means that even if a local network fails or is disrupted, the overall system performance is unaffected. Moreover, since the learning rate is updated through several independently running local networks, the architecture exhibits enhanced scalability and can more effectively detect novel malware, thereby improving the overall detection accuracy. The federated neural network operates as follows: initially, the dataset is partitioned into k subsets, where k corresponds to the number of slave nodes. Each subset is then allocated to a respective slave node. During each stage of training, every slave node classifies the images assigned by the master node and computes the error gradients. These gradients are then sent back to the master node, which updates the network weights accordingly. The updated weights are redistributed to the slave nodes, and this procedure continues until a specific number of iterations is achieved. The structure of the federated CNN network is illustrated in Figure 3.

DOI: 10.46121/pspc.53.4.20

Power System Protection and Control
ISSN:1674-3415

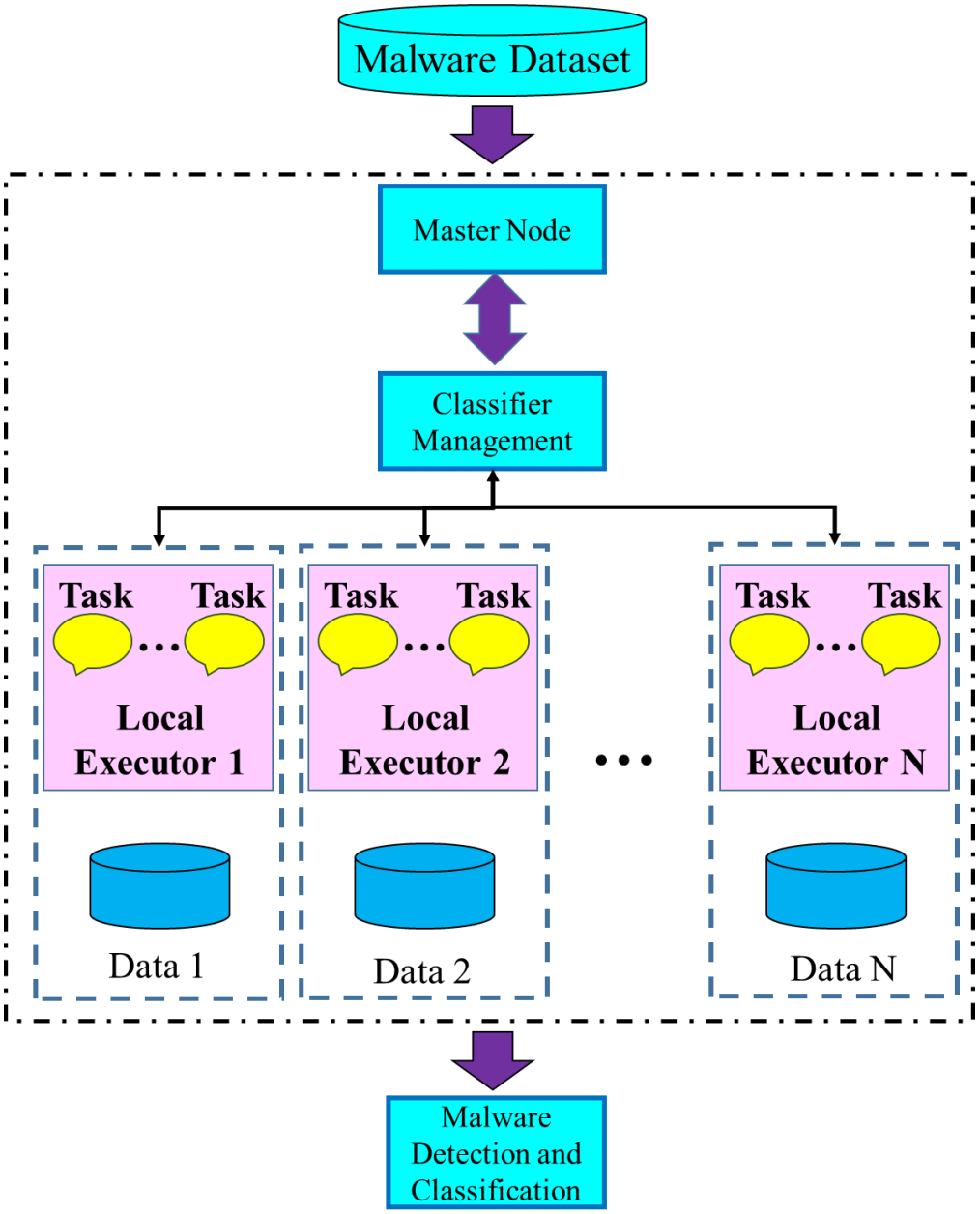

Figure 3. Architecture of the Federated CNN Network

3.3. Model Training Using SAM Algorithm

In this work, the training process for the CNN-based Federated Learning model is enhanced using the Sharpness-Aware Minimization (SAM) optimization algorithm. SAM is designed to improve the generalization capability of neural networks in non-IDD data environments by steering model updates toward flatter regions of the loss landscape during training. This approach leads to improved stability and accuracy of the model.

SAM has demonstrated superior performance on non-IID datasets by reducing overfitting and enabling the model to learn more robust parameters in the presence of heterogeneous data distributions. Unlike synchronous approaches (e.g., Synchronous SGD or SSGD), where failure of any client can halt the entire training process, in the proposed asynchronous training framework, each client trains independently. As a result, even if a client fails, the training process continues with the remaining clients. This asynchronous design improves fault tolerance and reduces the overall training time. Figure 4 demonstrates the update mechanism of the Federated CNN model parameters.

DOI: 10.46121/pspc.53.4.20

Power System Protection and Control
ISSN:1674-3415

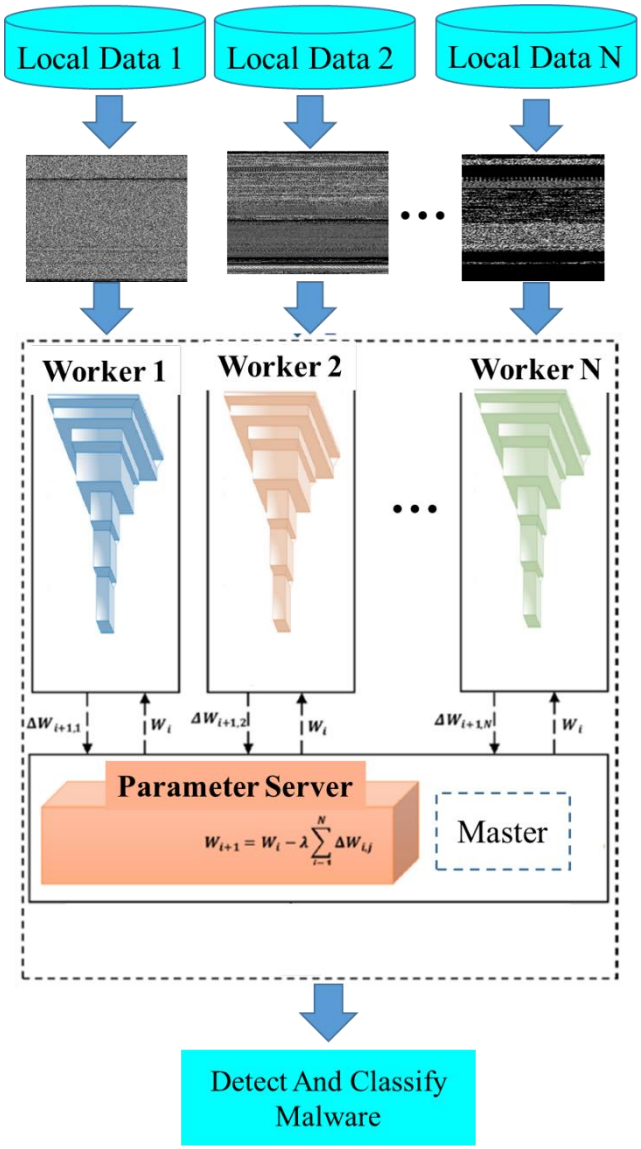

Figure 4. Federated CNN model's parameter updating procedure

Every worker node interacts with the central parameter server, as illustrated in Figure 4. All updates (including gradients and learning rate adjustments) are sent from the clients to the server, where they are aggregated. In general, weight updates in a federated system are computed as:

$$W_{i+1} = W_i - \lambda \sum_{j=1}^{N} \Delta W_{i,j} \quad (6)$$

Where, λ is a scaling factor and $\Delta W_{i,j}$ is the update vector from client j at i iteration.
In this work, SAM is used to perform local updates on each client through a two-step optimization process: First, a perturbation vector ϵ is computed to maximize the loss in the neighborhood of the current weights:

$$\epsilon = \rho \frac{\nabla L_j(W_i)}{\|\nabla L_j(W_i)\|} \quad (7)$$

Where, ρ is a small constant controlling the perturbation magnitude. Then, the final gradient is computed using the perturbed weights $(W_i + \epsilon)$:

$$\Delta W_{i,j} = \nabla L_j(W_i + \epsilon) \quad (8)$$

Finally, the local parameter update at client j is:

$$W_{i+1,j} = W_i - \alpha \Delta W_{i,j} \quad (9)$$

DOI: 10.46121/pspc.53.4.20

where α denotes the learning rate, and L_j is the loss function for client j. These updates are not applied directly to the global parameter vector; instead, they are first aggregated on the central server and then reflected in the global model. The key idea of SAM is that, instead of following the gradient in the direction of the current sharp minimum, each client moves toward a direction that considers loss sharpness in the surrounding region. This significantly improves the model's robustness to noisy and non-IID data and enhances the overall performance.
Algorithm 1 illustrates the update process of the local models at the worker nodes using the SAM algorithm. Algorithm 2 presents the update mechanism of the central server based on this algorithm in master node.

Algorithm 1: Weight Update Using the SAM Algorithm on the Client Side in Local Networks

Input: Clients E {e_1.....e_n}, Architecture of Base Convolutional Network, Initial Network weights, Control parameter ρ. Outcomes: Partial sharpness-aware gradient ΔW_i
1. For any client e_i in E apply 2. Load current parameters Wi from Global Server 3. Choose a uniformly distributed random sample s ∈ {1, ..., b} from local database 4. Calculate standard gradient $\nabla L_j(W_i)$ 5. Compute perturbation vector: $\epsilon = \rho \frac{\nabla L_j(W_i)}{\|\nabla L_j(W_i)\|}$ 6. Compute sharpness-aware gradient at $(W_i + \epsilon)$: $\Delta W_{i.j} = \nabla L_j(W_i + \epsilon)$ 7. Transfer the partially computed gradient $\Delta W_{i.j}$ to the Global Server 8. End

Algorithm 2: Weight Update in the Global Model

Input: Clients E {e_1.....e_n}, Architecture of Base Convolutional Network, Initial Network weights. Output: Global sharpness-aware gradient
1. Wait for deliver $\Delta W_{i.j}$ from any Client ei ∈ E 2. For each received $\Delta W_{i.j}$ do Accept the sharpness-aware gradient 3. Compute global gradient using: $W_{i+1} = W_i - \lambda \sum_{j=1}^{N} \Delta W_{i.j}$ 4. Update W_{i+1} and save in global server parameters 5. Transfer the updated global model W_{i+1} to all workers

3.4. Uncertainty Modeling Using MC-Dropout in the Federated Network

In the final stage of the proposed method, Monte Carlo Dropout (MC-Dropout) is employed to model decision-making uncertainty in the central model. This technique enables the estimation of a predictive distribution over the model's outputs without requiring complex Bayesian neural networks, making it particularly effective in federated learning environments, where data is often heterogeneous and unreliable. Unlike traditional dropout methods, where dropout is disabled during inference, MC-Dropout keeps the dropout mechanism active during test time. By performing multiple forward passes over the same input sample with different random dropout masks, the model can approximate a probabilistic distribution over the outputs.
Let x be the input data and $f_W(x)$ the output of the model, for a model using dropout, the final prediction is obtained by averaging the outputs from T stochastic forward passes:

$$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} f_{W_t}(x) \quad (10)$$

Where:

- T is the number of forward passes,
- $f_{W_t}(x)$ is the model output during the t-th pass with a random dropout.

To quantify the model's uncertainty, either the variance of the predictions or the entropy of the output distribution can be used. The predictive variance is calculated as:

$$Var(\hat{y}) = \frac{1}{T}\sum_{t=1}^{T}(f_{W_t}(x) - \hat{y})^2 \quad (11)$$

DOI: 10.46121/pspc.53.4.20

If the model uses a softmax output layer, the entropy of the predicted distribution can also serve as an uncertainty measure:

$$H(\hat{y}) = -\sum_{c=1}^{C} \hat{y}_c \log(\hat{y}_c) \quad (12)$$

Where C is the number of classes, $\hat{y}_c$ is the predicted probability for class c.

After performing MC-Dropout and computing either the variance or the entropy of the model's predictions, the system becomes capable of identifying input samples with high uncertainty. In other words, if a sample exhibits high predictive variance or entropy, it implies that the model is less confident about its decision. Integrating MC-Dropout into the central model of the federated architecture not only helps to preserve prediction accuracy, but also enhances the reliability of the decision-making system. Algorithm 3 outlines the process of uncertainty modeling using MC-Dropout in the federated learning system.

Algorithm 3: Pseudocode for Modeling Decision-Making Uncertainty Using MC-Dropout in a Federated Network

Inputs:
- Central CNN model with dropout layers: $f_W(x)$
- Input sample: x
- Number of stochastic forward passes: T
- Number of classes: C

Outputs:
- Predictive mean: ŷ
- Predictive variance: Var(ŷ)
- Predictive entropy: H(ŷ)

1. Initialize list of predictions: Y = []
2. For t = 1 to T do
3. Enable dropout during inference
4. Sample output from model: $y_t = f_W(x)$ // Stochastic forward pass
5. Append y_t to Y
6. End For

7. Compute predictive mean:
$\hat{y} = \frac{1}{T}\sum_{t=1}^{T} f_{W_t}(x)$

8. Compute predictive variance:
$Var(\hat{y}) = \frac{1}{T}\sum_{t=1}^{T}(f_{W_t}(x) - \hat{y})^2$

9. Compute predictive entropy:
$H(\hat{y}) = -\sum_{c=1}^{C} \hat{y}_c \log(\hat{y}_c)$

10. Return $\hat{y}, Var(\hat{y}), H(\hat{y})$

4. Experimental Results

The findings of the simulation of the suggested malware detection approach are shown in this section. For all simulations, the Python programming environment was used. The Malimg malware image dataset was employed for simulation purposes, with 70% of the data used for training and 30% used as the test set. To assess the effectiveness of the proposed approach, Accuracy, Precision, Recall, and F1-score are employed.

4.1. Dataset

This study made use of the Malimg dataset. This dataset is publicly available and serves as a baseline for malware classification on Kaggle. It includes 9,435 executable malware files that are divided into 25 malware families. Using nearest-neighbor interpolation, these malware files have been transformed into 32x32 grayscale pictures. Example images from three different malware classes are shown in Figure 5. Additionally, Table 1 lists all 25 malware classes along with their corresponding families.

DOI: 10.46121/pspc.53.4.20

Power System Protection and Control
ISSN:1674-3415

Table 1. Malware Classes in the Malimg Dataset

No	Class	Family	No	Class	Family	No	Class	Family
C1	Worm	Allaple L	C10	TDownloader	Swizzot.gen!I	C19	Dialer	Dialplatform B
C2	Worm	Allaple A	C11	TDownloader	Swizzor.gen!E	C20	TDownloader	Dontovo A
C3	Worm	Yuner A	C12	Worm	VB.AT	C21	TDownloader	Obfusca-tor.AD
C4	PWS	Lolyda AA 1	C13	Rogue	Fakerean	C22	Backdoor	Agent.FYI
C5	PWS	Lolyda AA 2	C14	Trojan	Alueron.gen!J	C23	Worm AutoIT	Autorun K
C6	PWS	Lolyda AA 3	C15	Trojan	Malex.gen!J	C24	Backdoor	Rbot!gen
C7	Trojan	C2Lop.P	C16	PWS	Lolyda AT	C25	Trojan	Skintrim N
C8	Trojan	C2Lop.gen!g	C17	Dialer	Adialer.C			
C9	Dialer	Instantaccess	C18	TDownloader	Wintrim BX			

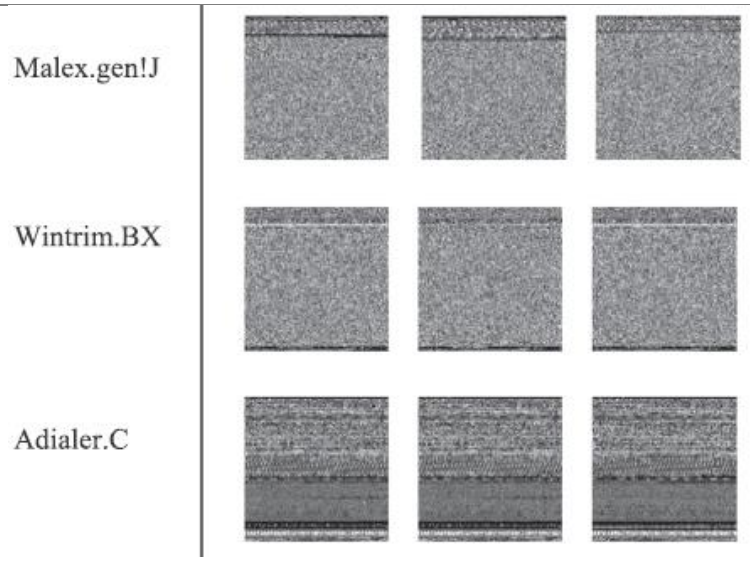

Figure 5. Example samples from three malware classes in the Malimg dataset

4.2. Evaluation Metrics

To evaluate the performance of the proposed method, the metrics Accuracy, Precision, Recall, and F1-score were used. The calculation formulas for these metrics are shown in Equations (13) to (16):

$$Accuracy = \frac{(TP+TN)}{(TP+FP+TN+FN)} \quad (13)$$

$$Precision = \frac{TP}{(TP+FP)} \quad (14)$$

$$Recall = \frac{TP}{(TP+FN)} \quad (15)$$

$$F1\ score = \frac{2*(Recall*Precision)}{(Recall+Precision)} \quad (16)$$

In the above equations, TP (True Positive) denotes the number of correctly identified positive cases, TN (True Negative) the number of correctly identified negative cases, FP (False Positive) the number of incorrectly identified positive cases, and FN (False Negative) the number of incorrectly identified negative cases.

4.3. Evaluation of the Proposed Method's Performance

In this section, the effectiveness of the presented system is evaluated utilizing various assessment metrics, focusing on the learning process, detection accuracy, the quality of the classification, and metrics related to uncertainty estimation. Additionally, the efficacy of the suggested model is contrasted with various methods.

4.3.1. Evaluation of the Training Process

The loss function trend of the proposed approach over 40 training epochs is illustrated in Figure 6. As observed in the graph, the loss value significantly decreases during the initial stages of training. Specifically, within the first five epochs, the loss drops sharply from approximately 1.6 to below 0.1. This rapid decline indicates the model's effective and fast learning capability in extracting initial patterns from the input data. As the training progresses, the downward slope of the loss curve gradually flattens, and the model approaches convergence. After approximately 15 epochs, the loss reduction becomes minimal, oscillating around a value close to zero, indicating

DOI: 10.46121/pspc.53.4.20

Power System Protection and Control
ISSN:1674-3415

that the model has reached the saturation point of learning and its parameters have stabilized. Since the final loss value is nearly zero, it can be concluded that the proposed model possesses a high learning capacity and is well-adapted to the training data. Moreover, the absence of significant fluctuations or increases in the loss demonstrates the model's stability and suggests that it does not suffer from overfitting.

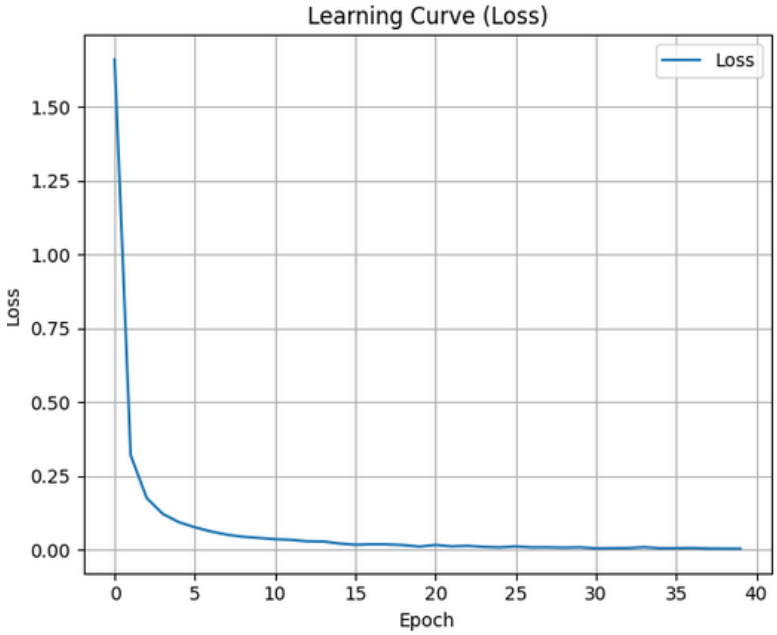

Figure 6. Learning curve based on the loss function

4.3.2. Evaluation of the Proposed Method in Terms of Receiver Operating Characteristic (ROC)

In machine learning, the performance of a binary classifier across various threshold values can be evaluated using the Receiver Operating Characteristic (ROC) curve, which is plotted based on two key metrics: True Positive Rate (TPR) and False Positive Rate (FPR). Figure 7 presents the ROC curve of the proposed malware detection method by plotting TPR versus FPR for different decision thresholds. An ideal classifier will have an ROC curve that leans toward the top-left corner, indicating a high TPR and low FPR. Conversely, a poor classifier will show a curve near the bottom-right corner, reflecting a low TPR and high FPR. As shown in the figure, the ROC curve for the proposed method is close to the top-left corner, which indicates that the classifier achieves a high TPR and low FPR. Furthermore, the Area Under the Curve (AUC) for all classes—except for two—was found to be very close to 1, reinforcing the effectiveness of the method.

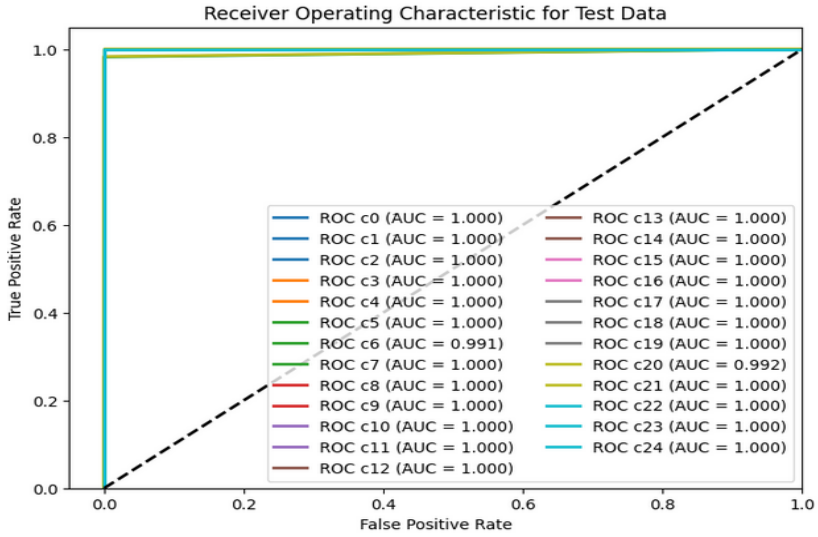

Figure 7. ROC curve of the proposed method

DOI: 10.46121/pspc.53.4.20

4.3.3. Comparison of the Results

Based on the Accuracy measure, Table 2 compares the suggested approach with a number of previous research. As can be observed, the proposed method outperforms all other approaches and achieves the highest accuracy in malware detection. Specifically, the accuracy of the proposed model reaches 99.86%, which surpasses that of other models such as MDC-RepNet (99.57%) and the hybrid ResNet50-VGG16 model (99.28%). Furthermore, more traditional models such as Random Forest and Autoencoder achieve lower accuracies of 98.68% and 96.22%, respectively. These results demonstrate that the carefully designed architecture of the proposed model plays a crucial role in enhancing its ability to learn discriminative features of malware samples. Therefore, it can be concluded that the suggested method achieves remarkable accuracy, possesses significant potential for real-world applications in cybersecurity and threat detection.

Table 2. Comparison of results in terms of Accuracy metric

Author	Method	Accuracy
XIAOFEI XING et al. [19]	Auto Encoder	96.22
Atitallah et al. [20]	Random Forest	98.68
Behera et al. [21]	ResNet50-VGG16	99.28
Li et al. [22]	MDC-RepNet[1]	99.57
-	**Presented**	99.86

A comparative analysis based on three widely-used evaluation metrics: Precision, Recall, and F-score is presented in Table 3. These metrics are essential for evaluating classification systems, especially in the domain of malware detection. According to the results, the proposed method significantly outperforms competing models across all three metrics. Specifically, the model achieves 99.87% Precision, 99.86% Recall, and 99.86% F-score. The values indicate that the system is not only highly accurate in correctly identifying positive samples (Precision), but also demonstrates strong sensitivity in detecting all actual positive instances (Recall). Additionally, the high F-score confirms a well-balanced performance between precision and recall.

Compared to MDC-RepNet, which is the closest in terms of performance (Precision: 99.50%, Recall: 99.52%, F-score: 99.51%), the proposed model provides noticeably higher precision and confidence. Other methods such as ResNet50-VGG16 and Random Forest, while performing reasonably well, still fall short in all three metrics.

These findings confirm that the designed architecture of the proposed model has been highly effective in extracting distinctive malware features and classifying them with high precision and completeness. This advantage provides the proposed method with a competitive edge over prior models, making it more suitable for deployment in sensitive and security-critical applications.

Table 3. Comparison of results in terms of Precision, Recall, and F-score metrics

Method	Precision	Recall	F-score
Auto Encoder	96.19	96.21	96.20
Random Forest	98.70	98.64	98.67
ResNet50-VGG16	99.00	99.00	99.00
MDC-RepNet	99.50	99.52	99.51
Presented	99.87	99.86	99.86

4.3.4. Evaluation of Proposed Method Performance in Terms of Uncertainty

Figure 8 presents a box plot illustrating the uncertainty distribution across 25 different malware classes, derived using the Monte Carlo Dropout (MC-Dropout) technique. The x-axis (Class Index) represents the malware class numbers (1 to 25), while the y-axis (Standard Deviation of Predictions) indicates the average standard deviation of the model's predictions over 100 Monte Carlo iterations. This standard deviation serves as a measure of the model's uncertainty: higher standard deviation corresponds to higher uncertainty in the model's predictions for a given class. In most classes, the standard deviation remains very low (ranging from near zero up to approximately 0.0007), which indicates a high level of model confidence in its classification outcomes. The lower edge of each box in the plot represents the first quartile (Q1) or 25th percentile, and the upper edge indicates the third quartile

[1] Structural Reparameterization and Multi-Scale Deep Convolutional Classifier Network

DOI: 10.46121/pspc.53.4.20

(Q3) or 75th percentile of the calculated variances. The interquartile range (IQR) thus covers 50% of the prediction variances.

For all classes, the box heights are consistently below 0.0003, further confirming the low levels of uncertainty. Additionally, the median line (in red) is positioned at or near the bottom of each box, and there are no lower whiskers, implying that the majority of standard deviation values are tightly concentrated near zero. In other words, the central 50% of the prediction variances are heavily skewed toward the lower end, with the median nearly coinciding with Q1. This indicates that the model made highly confident predictions with negligible uncertainty for most instances in each class. When examining uncertainty across individual classes, it is evident that classes 1, 2, 3, 9, 10, 16, 17, 18, 22, and 23 exhibit especially low and stable uncertainty, reflecting the model's greater robustness in classifying these categories.

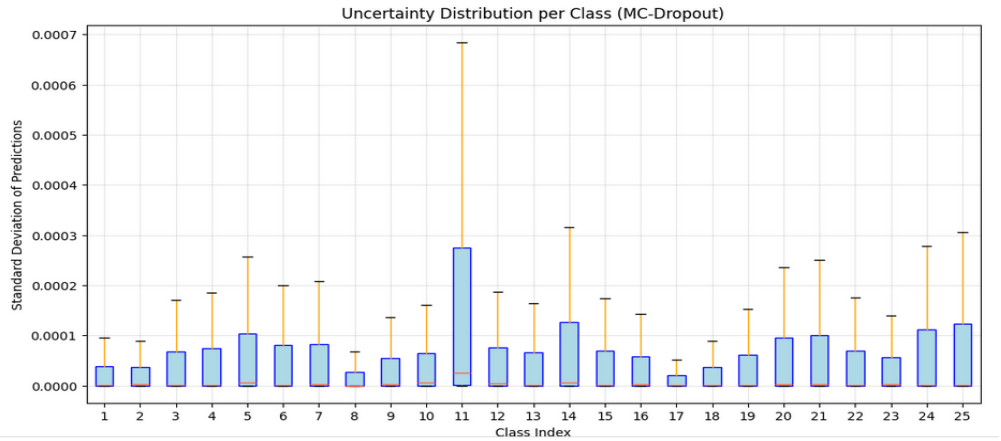

Figure 8. Uncertainty distribution in malware classification per class (based on MC-Dropout)

Figure 9 presents the uncertainty distribution over the test dataset. The x-axis shows the mean standard deviation of predictions for each malware image, serving as an indicator of the model's uncertainty per predicti
on. The y-axis represents the frequency, i.e., the number of test samples falling within each uncertainty range.
As clearly depicted in the histogram, the distribution is heavily concentrated around extremely low standard deviation values (close to zero). Specifically, nearly 2,300 test samples have a mean prediction standard deviation near zero. This sharp peak near zero indicates that for the vast majority of predictions, the model exhibited minimal uncertainty, hence demonstrating high confidence and consistency. The dense clustering at the lower end of the uncertainty spectrum strongly supports the effectiveness and reliability of the proposed method in producing robust and stable predictions. Only a small number of samples exhibit uncertainty in the range of 0.005 to 0.01, which still reflects high model confidence. Overall, the histogram offers strong evidence that the model is capable of effectively quantifying its uncertainty and maintains very high confidence across most of its prediction tasks.

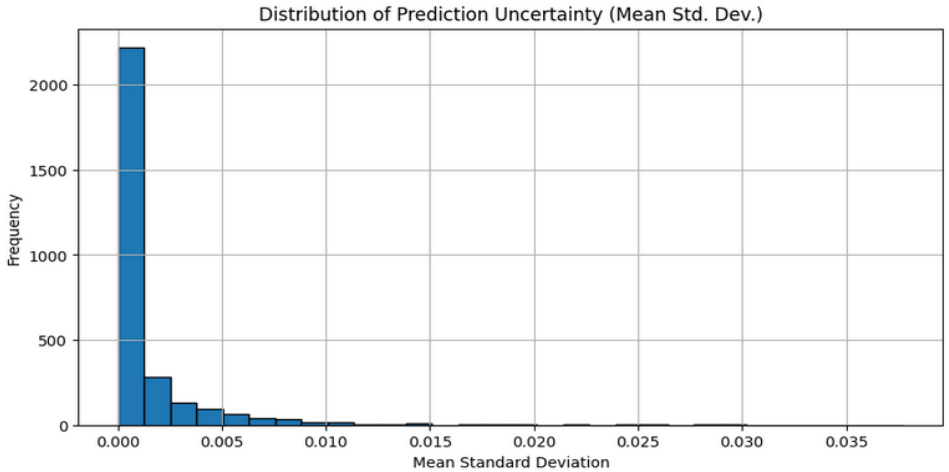

Figure 9. Uncertainty distribution in malware detection using the test dataset

DOI: 10.46121/pspc.53.4.20

CONCLUSION

The paper presents a new federated learning framework that is specifically adapted to CNN-based malware detection, which sufficiently solves the problems of heterogeneous and decentralized data. The proposed system guides the local models into flatter minima by embracing sharpness-aware minimization (SAM) and it helps in overall generalization and the problem of overfitting and client drift that usually occurs in a non-IID. scenario. Also, the Monte Carlo Dropout (MC-Dropout) protocol can be incorporated to enable reliable uncertainty estimation, so that the system could provide itself and the user with the degree of confidence about the predictions, which is a highly-desirable property in cybersecurity. Scalability and the resilience of the system are also enhanced with the use of asynchronous and fault-tolerant update mechanism that allows the system to continue running throughout even in the event of possible client failure. Experimental assessments confirm that the suggested framework attains higher detection precision, as much as 99.86%, and offers certain reliability in uncertainty estimation, which bears witness to its applicability in real settings of malware detection. Overall, the combination of federated learning, sharpness-aware optimization, and uncertainty modeling presents an interesting path to establishing robust, privacy-protecting cybersecurity systems that can effectively work in a heterogeneous and potentially noisy environment.

REFERENCES

1. Dutta, N., Jadav, N., Tanwar, S., Sarma, H. K. D., Pricop, E., Dutta, N., ... & Pricop, E. (2022). Introduction to malware analysis. Cyber Security: Issues and Current Trends, 129-141.
2. Alenezi, M. N., Alabdulrazzaq, H., Alshaher, A. A., & Alkharang, M. M. (2020). Evolution of malware threats and techniques: A review. International journal of communication networks and information security, 12(3), 326-337.
3. Rodríguez, R. J., Ugarte-Pedrero, X., & Tapiador, J. (2022). Introduction to the special issue on challenges and trends in malware analysis. Digital Threats: Research and Practice (DTRAP), 3(2), 1-2.
4. Gorment, N. Z., Selamat, A., Cheng, L. K., & Krejcar, O. (2023). Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions. IEEE Access, 11, 141045-141089.
5. Bensaoud, A., Kalita, J., & Bensaoud, M. (2024). A survey of malware detection using deep learning. Machine Learning With Applications, 16, 100546.
6. Redhu, A., Choudhary, P., Srinivasan, K., & Das, T. K. (2024). Deep learning-powered malware detection in cyberspace: a contemporary review. Frontiers in Physics, 12, 1349463.
7. Akhtar, M. S., & Feng, T. (2022). Detection of malware by deep learning as CNN-LSTM machine learning techniques in real time. Symmetry, 14(11), 2308.
8. Pei, J., Liu, W., Li, J., Wang, L., & Liu, C. (2024). A review of federated learning methods in heterogeneous scenarios. IEEE Transactions on Consumer Electronics, 70(3), 5983-5999.
9. Hu, K., Gong, S., Zhang, Q., Seng, C., Xia, M., & Jiang, S. (2024). An overview of implementing security and privacy in federated learning. Artificial intelligence review, 57(8), 204.
10. Liberti, F., Berardi, D., & Martini, B. (2024). Federated learning in dynamic and heterogeneous environments: Advantages, performances, and privacy problems. Applied Sciences, 14(18), 8490.
11. Myakala, P. K., Jonnalagadda, A. K., & Bura, C. (2024). Federated learning and data privacy: A review of challenges and opportunities. International Journal of Research Publication and Reviews, 5(12), 10-55248.

DOI: 10.46121/pspc.53.4.20

12. Darwish, R., & Roy, K. (2025, February). Comparative Analysis of Federated Learning, Deep Learning, and Traditional Machine Learning Techniques for IoT Malware Detection. In 2025 IEEE 4th International Conference on AI in Cybersecurity (ICAIC) (pp. 1-10). IEEE.
13. Nobakht, M., Javidan, R., & Pourebrahimi, A. (2024). SIM-FED: Secure IoT malware detection model with federated learning. Computers and Electrical Engineering, 116, 109139.
14. Serpanos, D., & Xenos, G. (2023, September). Federated learning in malware detection. In 2023 IEEE 28th International Conference on Emerging Technologies and Factory Automation (ETFA) (pp. 1-4). IEEE.
15. Çıplak, Z., Yıldız, K., & Altınkaya, Ş. (2025). FEDetect: a federated learning-based malware detection and classification using deep neural network algorithms. Arabian Journal for Science and Engineering, 1-28.
16. Hawana, A., Hassan, E. S., El-Shafai, W., & El-Dolil, S. A. (2025). Enhancing malware detection with deep learning convolutional neural networks: Investigating the impact of image size variations. Security and Privacy, 8(2), e70000.
17. Zhang, D., Song, Y., Xiang, Q., & Wang, Y. (2025). IMCMK-CNN: a lightweight convolutional neural network with multi-scale kernels for image-based malware classification. Alexandria Engineering Journal, 111, 203-220.
18. Eftekhari, M., Yousef Sanati, M., & Mansoorizadeh, M. (2025). Malware detection using federated learning and incremental learning. Electronic and Cyber Defense, 13(1), 117-130.
19. Xing, X.; Jin, X.; Elahi, H.; Jiang, H.; Wang, G. A Malware Detection Approach Using Autoencoder in Deep Learning. IEEE 722 Access 2022, 10, 25696–25706.
20. Atitallah, S. B., Driss, M., & Almomani, I. (2022). A novel detection and multi-classification approach for IoT-malware using random forest voting of fine-tuning convolutional neural networks. Sensors, 22(11), 4302.
21. Behera, Geeta Gayatri, Jitesh Pradhan, and Alekha Kumar Mishra. " A Hybrid Deep Learning Malware Detection Model with The Fusion of VGG-16 and ResNet-50 Architectures." Procedia Computer Science 258 (2025): 1659-1668.
22. Li, Sicong, et al. "A lightweight model for malicious code classification based on structural reparameterisation and large convolutional kernels." International Journal of Computational Intelligence Systems 17.1 (2024): 30.
atures-to-make-driving-safer-and/}.

DOI: 10.46121/pspc.53.4.20