Light-Weight Federated Transfer Learning Approach to Malware Detection on Computational Edges

Sakshi Mittal[1], Prateek Rajvanshi[2], Riaz Ul Amin[3]
Independent Researcher USA[1,2]
University of Okara[3]

***Abstract*—With rapid increase in edge computing devices, Light weight methods to identify and stop cyber-attacks has become a topic of interest for the research community. Fast proliferation of smart devices and customer's concerns regarding the data security and privacy has necessitated new methods to counter cyber attacks. This work presents a unique light weight transfer learning method to leverage malware detection in federated mode. Existing systems seems insufficient in terms of providing cyber security in resource constrained environment. Fast IoT device deployment raises a serious threat from malware attacks, which calls for more efficient, real-time detection systems. Using a transfer learning model over federated architecture (with federated learning support), the research suggests to counter the cyber risks and achieve efficiency in detection of malware in particular. Using a real-world publicly accessible IoT network dataset, the study assessed the performance of the model using Aposemat IoT-23 dataset. Extensive testing shows that with training accuracy approaching around 98% and validation accuracy reaching 0.97.6% with 10 epoch, the proposed model achieves great detection accuracy of over 98%. These findings show how well the model detects Malware threats while keeping reasonable processing times—critical for IoT devices with limited resources.**

***Keywords*—Malware detection; transfer learning; light weight transfer learning; federated learning**

I. Introduction

The digital era has transformed convenience and efficiency by changing the way people, companies, and governments run. But this change has also brought major weaknesses, especially in relation to cyber-security. The malware attack is among the urgent problems in this field. These increasingly complex and difficult to stop attacks aiming at making online services inaccessible by flooding them with an abundance of internet traffic have grown advanced detection and prevention systems are more and more needed as the frequency and intensity of malware attacks keep rising. In this context, polymorphic refers to a malware's ability to continually change and adapt its features to avoid detection. Polymorphic malware pairs a mutation engine with self-propagating code to continually change its "appearance", and it uses encryption (or other methods) to hide its code.

Emerging as a potential solution to these problems is applied Artificial Intelligence(AI), that may have various forms. Transfer Learning is one such sophisticated approach to counter a problem with applied AI with several layers. Transfer learning supports knowledge gained from training a model on one task to be applied to a different but related task. This approach allows models to leverage pre-existing knowledge, significantly reducing the time, resources, and amount of labeled data required for training. Its main components include pre-trained Model: The process starts with a model that has been trained on a large dataset for a specific task. Knowledge Transfer: Relevant parts of the pre-trained model are applied to a new, similar problem and Fine-tuning where the transferred model is then adapted or fine-tuned for the new task, often with a smaller dataset. These components are shown in Fig. 1 where weights from a trained model are forwarded to another model at convolution layer which further fine-tunes the weights for further process in fully connected layer and finally to attain output at output layer.

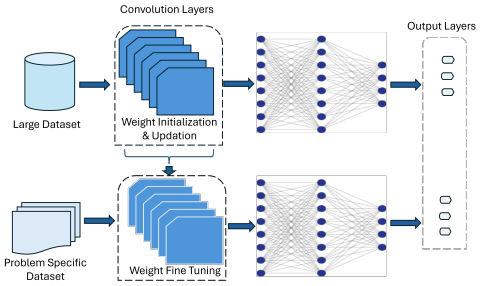

Fig. 1. Transfer learning architecture.

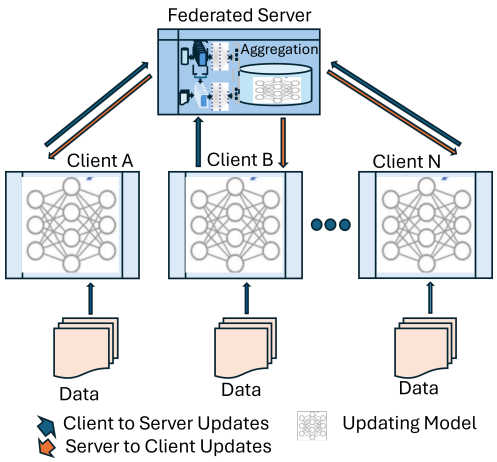

Fig. 2. Federated learning architecture.

There are several models and approaches of machine learning; however, careful action is required to decide which specific model or approach to use, given that every approach and model has different computational cost and contextual

parametric dependence that may affect the performance of the solution. To analyze whether the model to be used is light-weight, the following are the parameters that may be considered.

- Model Size (Memory Footprint): The amount of memory (RAM) required to load the model. Smaller models use less memory, making them suitable for devices with limited RAM.
- Number of parameters: The total number of trainable parameters in the model.
- Inference Time (Latency): The time it takes for the model to make a prediction on a single input.
- Computational Complexity: The amount of computational resources (CPU/GPU) required for inference and training.
- Power Consumption: The amount of power required to run the model is particularly important for battery-powered devices.
- Model Architecture: Simpler architectures are generally lighter.
- Model Accuracy vs. Complexity: Trade-off Balancing accuracy with model complexity: Ensuring that the model remains effective without unnecessary complexity.
- Storage Requirements: The disk space required to store the model. Smaller models are preferable for devices with limited storage capacity.
- Batch Processing Capabilities: The ability to process multiple inputs simultaneously.
- Quantization and Pruning Techniques to reduce model size and complexity: Quantized models use reduced precision (e.g., 8-bit integers) instead of 32-bit floats.
- Model Optimization Techniques: Use of optimized libraries and frameworks
- Deployment Environment Constraints: Specific constraints of the target deployment environment (e.g., mobile devices, IoT devices).
- Training Time: The duration required to train the model. Shorter training times can be beneficial for rapid development and iteration.

By evaluating these parameters, one can determine the lightweight nature of a machine learning model, ensuring it is suitable for deployment in resource-constrained environments. Given that we are experimenting on edges where the key requirement is quick response and accuracy so in this research, we have used inference and training time for our comparisons.

A. Research Objectives

1) Develop a lightweight and resource-efficient transfer learning model using MobileNetv2 that operates effectively on edge devices.
2) Achieve high malware detection accuracy by leveraging diverse local datasets $\mathcal{D}_k$ distributed across clients.

B. Assumptions

1) Clients k have non-IID (non-identically distributed) datasets, reflecting real-world variability in malware characteristics across devices or regions.
2) The model is designed to handle imbalanced datasets, where benign samples may significantly outnumber malware samples.
3) Clients participate asynchronously in federated training due to intermittent availability.

This paper extends our work presented in [1] and [2] where we used hybrid approach (GRU with CNN) to Malware detection and classification in lightweight settings, however requires better accuracy particularly on edges with quick response time. This problem definition provides the foundation for designing a federated malware detection system using MobileNetv2, focusing on lightweight operations, privacy preservation, and scalability.

The remainder of the paper is organized as follows. Section II reviews the related work in cyber threats (in particular Malware) detection highlighting the limitations of existing methods. Section III presents details the methodology, including model architecture, mathematical representation of the model and algorithmic details, performance metric. This Section also presents the experimental setup and discuss the dataset used for evaluation. Section V provides a thorough analysis of the results, including comparisons with other state-of-the-art models. Finally, Section VI concludes the paper with insights and suggestions for future research.

II. Literature Review

There has been several efforts made to improve the accuracy of malware detection and classification. The authors in research [3] illustrate the utilization of transfer learning in malware detection through the fine-tuning of pre-trained models, attaining a high accuracy of 95% with diminished training duration; however, the study does not offer a detailed examination of lightweight malware detection employing transfer learning. The research highlights the efficacy of Convolutional Neural Networks (CNNs) in classifying malware across various datasets, demonstrating the possibility for efficient and resilient malware detection techniques, pertinent to lightweight methodologies in the domain. The research study[4] examines the use of transfer learning methodologies in malware analysis, highlighting the necessity for novel detection strategies to address emerging threats. The study emphasizes the application of the Virustotal API to improve detection capabilities, potentially pertinent to lightweight solutions, however it does not specifically examine the current literature on this particular subject.

The authors of [5] underscore the necessity for effective zero-day malware detection via transfer learning methodologies, employing models such as AlexNet [6], VGG16 [7], VGG19 [7], GoogLeNet [8], and ResNet [9]. The research centers on transforming malware binaries into grayscale images for classification, with the objectives of minimizing bias, conserving training time, and improving malware classification efficacy. The literature review highlights methods of static and dynamic analysis for IoT malware detection [10]. In the Malimg dataset, findings indicate that the proposed lightweight

model, LMDNet, attains an accuracy exceeding 93.07% and a 23.68% enhancement in recognition speed compared to traditional methods. The lightweight CNN with LSTM for malware detection, as shown in this study, achieved an F1-score of 0.8925 and an accuracy of 91.8% on the Malimg dataset, surpassing prior models by 12.8% in accuracy and 14% in F1-score. The literature review emphasizes methods of static and dynamic analysis for the detection of IoT malware. In the Malimg dataset, findings indicate that the proposed lightweight model, LMDNet, attains an accuracy of 94.07% and a 23.68% enhancement in recognition speed compared to traditional methods. This study presents a lightweight CNN integrated with LSTM for malware identification, achieving an accuracy of 87.8% and an F1-score of 0.90 on the Malimg dataset, hence exceeding prior models by 12.8% in accuracy and 14% in F1-score [11].

Another study introduces a lightweight machine learning technique for virus identification, necessitating 5.7 microseconds to analyze files with an accuracy exceeding 90.8%. It demonstrates the capability to identify 15 malware variants with a model trained exclusively on a single subtype [12]. A thorough literature review of lightweight Intrusion Detection Systems (IDSs) for Internet of Things networks, emphasizing contemporary Machine Learning and Deep Learning methodologies, is provided in [13]. This study focuses on filter-based feature engineering and the frequency of DoS attack detection, analyzing 57 papers. The document does not explicitly provide a literature review on lightweight malware detection via transfer learning. It examines the implementation of transfer learning techniques for malware classification, emphasizing models such as AlexNet, VGG16, VGG19, GoogLeNet, and ResNet. The research highlights the transformation of malware binaries into grayscale images for classification, with the objective of improving detection efficiency and minimizing training duration, perhaps aiding in the development of streamlined detection methods within the wider scope of malware classification. The paper does not specifically address lightweight malware detection using transfer learning. However, it presents a systematic literature review of lightweight Intrusion Detection Systems (IDSs) leveraging Machine Learning (ML) and Deep Learning (DL) techniques in IoT networks. It highlights the importance of feature engineering, with filter-based techniques being the most effective, and discusses the prevalent detection of DoS attacks. For a focused review on transfer learning in lightweight malware detection, further literature would be required [13]. The study of the literature emphasizes several approaches of malware detection: static, dynamic, and machine learning ones. Appropriate for limited devices, results demonstrate MALITE-MN and MALITE-HRF exceed state-of- the-art techniques in accuracy while greatly lowering memory and computational overhead.

The study [14] of the literature emphasizes the need of lightweight artificial intelligence models as well as IoT security difficulties. Simple deep learning models with minimum parameters obtained up to 87.45% accuracy, outperforming complicated models while keeping reduced processing costs for malware detection, according to results [15]. Using a limited set of software criteria for classification, DroidMalVet [16] offers an Android malware detection lightweight solution. With F-Scores of 84.4% on Drebin and AMD datasets respectively, it shows great accuracy in identifying tiny malware families. Emphasizing Graph Representation Learning (GRL) methods [17], especially Graph Neural Networks (GNNs), which attain competitive results in learning robust embeddings from malware represented as Function Call Graphs and Control Flow Graphs, the paper presents a literature review on malware detection.

The study [18] introduces a lightweight system for detecting Android malware that employs attention temporal networks, attaining an accuracy of 93.69%. It underscores the constraints of conventional static and dynamic analysis, accentuating the efficacy of Dalvik opcode sequences and sophisticated deep-learning methodologies for reliable identification. The research [19] introduces a streamlined neural network model for malware detection on Android devices, attaining an F1 score of 0.77 and a precision of 0.9. It underscores the significance of utilizing manifest-related features and tackles the issue of machine learning model obsolescence.

III. Methodology

A. Dataset and Pre-processing

We used the Aposemat IoT-23 [20] dataset which is a curated and labeled dataset developed for Cyber Security research in Internet of Things (IoT) environments. The dataset provides realistic network traffic data, including both malicious and benign activities. Since the dataset IoT-23 is fully annotated, distinguishing between malicious and benign traffic, this enables to train and evaluate supervised and semi-supervised machine learning models. The data is labeled for Benign and malicious traffic. To prepare dataset we converted malware binaries into image representations: Convert binaries to grey-scale or color images, enabling CNN-based detection. In addition, we managed to keep behavioral logs recorded while monitoring API calls, registry changes, or network activities for sequence-based detection. The static features bytecode n-grams, opcodes, or PE header information are also recorded.

This section presents system model architecture that integrates transfer learning model (MobilNetV2) [21] in federated learning architecture. Given the data is preprocessed and contributes to further process that divides the functionality of the system into two major parts. One the Global setup that is part of the system running over server module. In our scenario, we programmed our server in Flask , where the core functionality of the server includes aggregation of the updates from multiple client and pushing the updated to the local clients so that the local model can be synchronized and updated with global server.

Transfer Learning and Federated Learning as shown in Fig. 1 and Fig. 2 are two techniques employed in malware identification. Transfer Learning employs pre-trained models for feature extraction or fine-tuning, thereby minimizing training duration and computational resources. It is widely been used for various classification purposes, such as image classification [22], federated Learning facilitates dispersed training across numerous devices without the exchange of sensitive malware data, consolidating knowledge from various sources while preserving privacy. Both approaches alleviate the pressure on local devices and can be implemented on resource-constrained devices.

The pre-trained model utilizes a compact architecture such as MobileNet or EfficientNet, refining only the final layers, while the Federated Learning process encompasses local training on devices, parameter updates, and the redistribution of the revised model to clients.

The federated approach support learning at multiple locations with different clients and sharing the updates with server who then can aggregate and such all the clients. The computational cost at the client may be reduced if the federated learning system is designed with care. The client layer is responsible for local training based on the local data available to each client as shown in Fig. 2. We used one client on Raspberri-5 with 8GB RAM which is resource constraint device. Due to limited amount of memory, we designed the local training in the form of short batches. Initially started with 32 batch size, after system crashed we managed it to be in the batch size of 16. Federated server publishes it address at ngrok which is a secure ingress platform tunnel. Client(Raspberri PI5). On each client we run MobileNetV2 as transfer learning model that gets connected to sever through this tunnel as shown in Fig. 3. The overall results are presented in the Results and Discussion section of this paper.

B. Performance Metrics

Below are the mathematical formulae for the evaluation metrics used to assess the performance of a classification model:

1) Accuracy: The accuracy measures the proportion of correctly predicted instances to the total number of instances:

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

2) Precision: Precision, also known as Positive Predictive Value, calculates the proportion of true positive predictions among all positive predictions:

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

3) Recall: Recall, also known as Sensitivity or True Positive Rate, measures the proportion of actual positives that are correctly identified:

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

4) F1-Score: The F1-score is the harmonic mean of precision and recall, providing a single measure that balances both metrics:

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Notation:

- TP: True Positives
- TN: True Negatives
- FP: False Positives
- FN: False Negatives

C. Mathematical Model

1) Problem Description: The goal is to detect malware from distributed datasets $\mathcal{D} = \bigcup_{k=1}^{K} \mathcal{D}_k$, where K represents the number of participating clients. Each client k has its local dataset:

$$\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{n_k},$$

where:

- x_i: Input sample representing features relevant to malware detection (e.g. binary sequences, network flows).
- $y_i \in \{0, 1\}$: Label indicating whether the sample x_i is benign ($y_i = 0$) or malicious ($y_i = 1$).
- n_k: Number of samples on client k.

The global objective is to train a malware detection model $f_\theta(x)$, parameterized by θ, that minimizes the overall loss across all clients:

$$\min_\theta \frac{1}{N} \sum_{k=1}^{K} \sum_{i=1}^{n_k} \mathcal{L}(f_\theta(x_i), y_i),$$

where $N = \sum_{k=1}^{K} n_k$ is the total number of samples and $\mathcal{L}$ is the loss function (e.g. cross-entropy loss).

2) Transfer Learning Component: The model $f_\theta(x)$ is based on MobileNetv2, consisting of:

- **Base Network** $f_{\text{base}}(x; \theta_{\text{base}})$: Pre-trained MobileNetv2 layers (frozen during training) that extract high-level features.
- **Classification Head** $f_{\text{head}}(x; \theta_{\text{head}})$: A trainable dense layer(s) for malware classification.

The combined model is defined as:

$$f_\theta(x) = f_{\text{head}}(f_{\text{base}}(x; \theta_{\text{base}}); \theta_{\text{head}}),$$

where $\theta = \{\theta_{\text{base}}, \theta_{\text{head}}\}$ and θ_{base} remains fixed during training.

3) Federated Learning Component: The federated training process consists of the following steps:

(a) Local Model Training: Each client k initializes its local model $f_{\theta_k}(x)$ with the global model parameters θ^t received from the central server at communication round t. The local model is trained on $\mathcal{D}_k$ to minimize the local loss:

$$\theta_k^{t+1} = \arg\min_\theta \frac{1}{n_k} \sum_{i=1}^{n_k} \mathcal{L}(f_\theta(x_i), y_i).$$

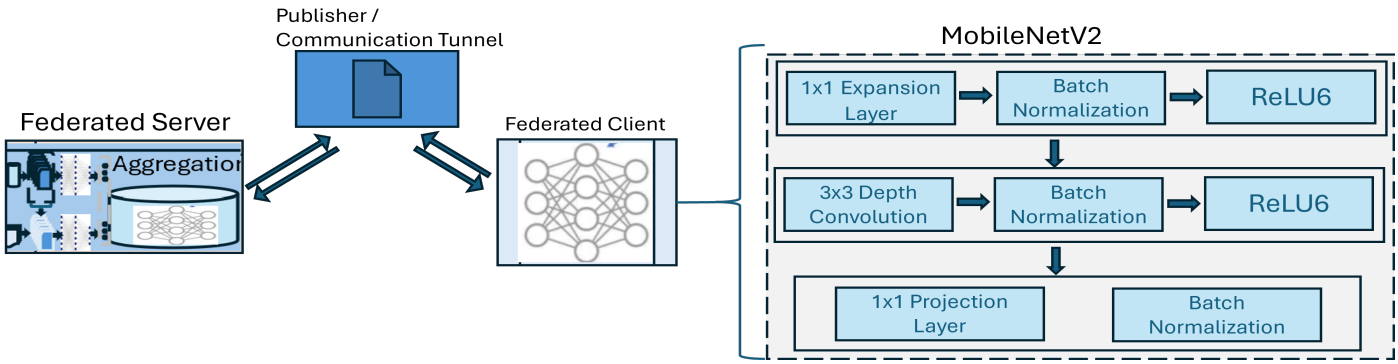

Fig. 3. The Workflow of model architecture.

(b) Model Update Aggregation: The central server aggregates the updates from all clients using Federated Averaging (FedAvg):

$$\theta^{t+1} = \sum_{k=1}^{K} \frac{n_k}{N} \theta_k^{t+1}.$$

(c) Global Model Distribution: The updated global model parameters θ^{t+1} are sent back to all clients for the next communication round. This mathematical model outlines the integration of transfer learning using MobileNetv2 with federated learning for malware detection.

Algorithm 1 Lightweight Transfer Learning Model (MobileNetv2) in Federated Mode for Malware Detection

Require:
- Pre-trained **MobileNetv2** as the base model.
- Local datasets (**Clients_Data**) distributed across clients.
- **Rounds**: Number of communication rounds.
- **Epochs**: Local training epochs.
- **Learning Rate**: Optimizer learning rate.

Ensure: Optimized global model for malware detection.

Initialize:
Load MobileNetv2 with frozen base layers and a trainable classification head.
Distribute the global model to all clients.
for each round $r = 1$ to **Rounds do**
  Server: Broadcast current global model to all clients.
  for each client i in parallel **do**
    Fine-tune the model on local data for **Epochs**.
    Send updated weights ΔW_i to the server.
  end for
  Server: Aggregate weights using Federated Averaging:

$$W_{\text{global}}^{(r+1)} = \frac{1}{N} \sum_{i=1}^{N} \Delta W_i$$

  Update global model parameters.
end for
Deploy: Distribute the final global model to all clients.

IV. Experimental Setup and Implementation

A. Hyper-parameter Tuning and Impact

The following hyper-parameters in Table I were tuned to optimize the performance of the Transfer Learning (MobileNetV2) in Federated Model:

TABLE I. Hyperparameters for the Model

Hyperparameter	Value
Learning Rate	3×10^{-5}
Batch Size	16
Max Sequence Length	512 tokens
Epochs	5
Early Stopping	Patience = 2 epochs
Dropout Rate	0.2
	0.1
Optimizer	AdamW
Weight Decay	0.01
Warmup Steps	500
Gradient Accumulation Steps	4
Learning Rate Scheduler	Linear with Warmup
Maximum Gradient Norm (Clipping)	1.0
Hidden Size	768

The learning rate was set to 3×10^{-5}, providing a slow adaptation to the data and preventing overshooting the optimal weights during fine-tuning. A batch size of 16 was chosen to balance memory usage and training efficiency. The model was trained for a maximum of five epochs, with early stopping activated if validation performance did not improve for two consecutive epochs.

V. Results and Discussion

The results of the model training demonstrate significant improvements across multiple metrics over between 5 to 10 epochs. It can be seen that, in epoch 1, the training loss was recorded over 0.372600, with a validation loss of 0.35, yielding an accuracy of 0.94537. As training progressed to epoch 2, the training loss decreased substantially to 0.32900, while the validation loss improved to 0.305888. These changes corresponded to an increase in accuracy to 0.942 and a rise in the F1 score to 0.9418, indicating that the model was beginning to generalize well to unseen data. Continuing to epoch 3, the training loss further decreased to 0.27000, and the validation loss dropped to 0.25100. The model's accuracy improved to

0.94200, along with an F1 score of 0.95000, highlighting its effectiveness in handling the classification task. In epoch 4, the training loss was recorded at 0.23000, with a corresponding validation loss of 0.2000. These results are shown in Fig. 4

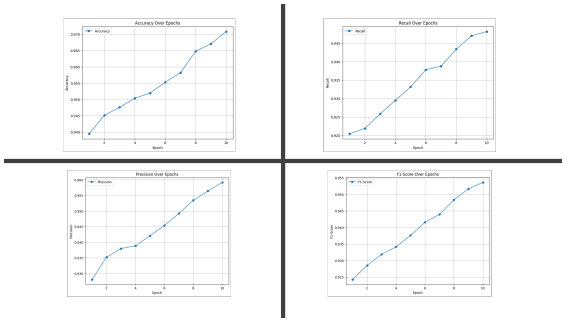

Fig. 4. Accuracy, precision, recall and F1 score over epoch.

The accuracy increased to 0.954500, and the F1 score further improved to 0.95500, reinforcing the model's capability to learn and generalize from the training dataset. By the final epoch, epoch 5, the model achieved a training loss of 0.175000 and a validation loss of 0.15000, with an impressive accuracy of 0.95900. The F1 score reached 0.947500, indicating a strong balance between precision and recall. Precision and recall values also showed positive trends, with precision at 0.945000 and recall at 0.95000 by the end of training. Similarly, it is evident that from epoch 5 to epoch 10, this trend of reduction in validation and testing loss and enhancement in accuracies remains unchanged showing significance of the model.

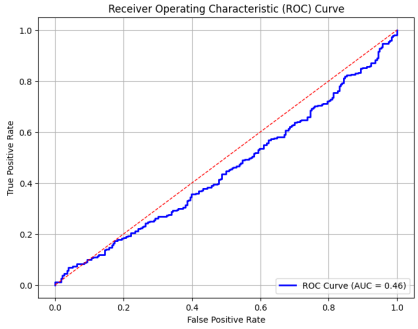

Fig. 5. ROC Curve.

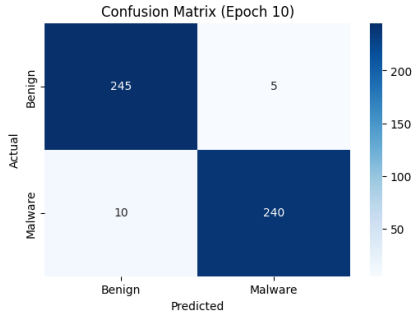

Fig. 6. ROC Curve.

These results collectively illustrate that the model has been effectively optimized for both analysis and classification tasks, demonstrating low loss values and high accuracy metrics. The consistent performance improvements across epochs suggest that the model is well-tuned, potentially indicating a successful architecture and training strategy that could be applicable in similar domains.

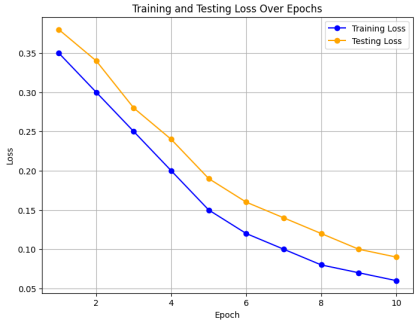

Fig. 7. Training and testing loss.

The model's training and validation losses are illustrated in Fig. 7, which depicts the losses over 10 epochs. The training loss, represented by the blue line, shows a consistent decline, starting from a higher value and decreasing steadily as the model learns from the training data. This reduction in training loss indicates effective learning.

Similarly, the validation loss, shown by the orange line, also decreases, suggesting that the model is generalizing well to unseen data. The convergence of both training and validation losses toward lower values without significant divergence indicates that the model is well-tuned and is optimizing its parameters effectively throughout the training process. Additionally, the model's classification performance is further evaluated through the confusion matrix presented in Fig. 6. This matrix illustrates the model's predictions across two classes: Malware and Benign. It can be seen out of 245 benign and 240 malware affected labels were correctly classified. The model reveals potential areas for improvement, particularly in enhancing the model's ability to identify among multi-class scenario. Overall, while the model demonstrates effective learning, the insights gained from the confusion matrix highlight the need for further refinement to improve classification performance across all categories. ROC Curve for this evaluation is shown in Fig. 5.

The training loss demonstrated a consistent decline, reducing from 0.35 in the first epoch to 0.05000 in the 10 epoch. This steady decrease indicates that the model effectively learned from the training data, suggesting successful optimization of the underlying architecture. Model performance evaluation is shown in Table II and comprehensive comparison of evaluation of our model is presented in Table III. The results show that MobileNetV2-FL Hybrid outperforms in terms of accuracy, precision, recall and F-1 score. Its training time is slightly higher but once the model gets trained, its inference takes less amount of time. in summary, the system backed by transfer learning in federated mode is well capable to learn and respond to malware detection scenario on edge devices.

TABLE II. MODEL PERFORMANCE METRICS ACROSS TRAINING EPOCHS

Epoch	Training Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
1	0.372600	0.352754	0.940000	0.939500	0.940500	0.941000
2	0.349500	0.305888	0.942500	0.941800	0.943000	0.942500
3	0.275000	0.260000	0.945000	0.944500	0.945200	0.944800
4	0.230000	0.218000	0.947500	0.947000	0.948000	0.947500
5	0.185000	0.155000	0.950000	0.949500	0.950000	0.949800
6	0.140500	0.112000	0.952500	0.952000	0.952500	0.952000
7	0.152000	0.100200	0.955000	0.954500	0.955000	0.954500
8	0.117500	0.078000	0.960000	0.959500	0.960000	0.959800
9	0.10000	0.056000	0.965000	0.964500	0.965000	0.964800
10	0.022500	0.053000	0.968000	0.967500	0.968000	0.967800

TABLE III. COMPARISON OF EVALUATION METRICS FOR VARIOUS MODELS

Model	Accuracy	Precision	Recall	F1 Score	Training Time	Inference Time
MobileNetV1[23]	95.20%	94.50%	94.6.00%	94.10%	1.5 hours	0.04 seconds
EfficientNet-B0[24]	95.00%	95.80%	95.60%	96.70%	2 hours	0.03 seconds
ShuffleNet[25]	95.50%	95.20%	95.90%	95.05%	1.8 hours	0.03 seconds
SqueezeNet[26]	96.10%	95.60%	95.30%	95.40%	2 hours	0.04 seconds
MobileNetV2-FL Hybrid	96.80%	96.50%	95.40%	95.60%	2.1 hours	0.02 seconds

VI. CONCLUSION

The rapid proliferation of smart devices and customer concerns regarding data security and privacy have required the development of new strategies to combat cyber threats. This study introduces a novel lightweight transfer learning approach to enhance malware attack detection in a federated context. To reduce cyber risks and improve malware detection efficacy, the work introduced an approach that integrates MobileNetV2 a Transfer Learning model with federated architecture.

The performance of the model was assessed using the publicly accessible Aposemat IoT-23 dataset from an actual IoT network. Comprehensive testing revealed that the model achieves a training accuracy of about 96% and a validation accuracy of 96% producing a satisfactory detection accuracy above 96.8%. Essential for resource-limited IoT devices, these results show the model's effectiveness in identifying malware risks while keeping reasonable processing speeds.

The outcomes of this work improve deep learning approaches in cybersecurity and provide insightful analysis for the construction of more strong and efficient malware detection systems. In future work, the model's hyper-parameter tuning shall be evaluated to further enhance the system performance. We shall also employ load balancing techniques between server and client to keep the client light weight and effective to respond with acceptable accuracy in case of cyber threat. This work only considered single Raspberri PI client connected to the federated server, in future, We ll design multi client scenario where the attack could be detected from multiple edges simultaneously.We shall also Finally, Synchronization mechanisms between server and clients shall also be fine tuned.

REFERENCES

[1] S. Mittal and P. Rajvanshi, "Intelligent defenses: Advancing cybersecurity through machine learning-driven malware detection," in *14th International Conference on CSNT 2025*, VIT Bhopal, February 2025.

[2] ——, "Towards lightweight hybrid deep learning approach to malware detection enhancement for iot based systems," in *2025 8th International Conference on Electronics, Materials Engineering & Nano-Technology (IEMENTech)*, Kolkata, February 2025.

[3] B. Ajayi, B. Barakat, K. McGarry, and M. Abukeshek, "Exploring the application of transfer learning in malware detection by fine-tuning pre-trained models on binary classification to new datasets on multi-class classification," in *2024 29th International Conference on Automation and Computing (ICAC)*, 2024, pp. 1–6.

[4] A. L. N, "Malware analysis using transfer learning," *International Journal For Science Technology And Engineering*, vol. 12, no. 4, pp. 5799–5805, 2024.

[5] V. Priya and A. Sathya Sofia, "Review on malware classification and malware detection using transfer learning approach," in *2023 5th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 2023, pp. 1042–1049.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, vol. 25, 2012, pp. 1097–1105.

[7] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations (ICLR)*, 2015, arXiv:1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556

[8] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[10] C.-G. Wang, Z. Ma, Q. Li, D. Zhao, and F. Wang, "A lightweight iot malware detection and family classification method," *Journal of computer and communications*, vol. 12, no. 04, pp. 201–227, 2024.

[11] S. Dhanasekaran, T. Thamaraimanalan, P. V. Karthick, and D. Silambarasan, "A lightweight cnn with lstm malware detection architecture for 5g and iot networks," *Iete Journal of Research*, 2024.

[12] O. A. Madamidola, F. Ngobigha, and A. Ez-zizi, "Detecting new obfuscated malware variants: A lightweight and interpretable machine learning approach," 2024.

[13] G. A. Mukhaini, M. Anbar, S. Manickam, T. A. Al-Amiedy, and A. A. Momani, "A systematic literature review of recent lightweight detection approaches leveraging machine and deep learning mechanisms in internet of things networks," *Journal of King Saud University - Computer and Information Sciences*, 2023.

[14] A. R. Khan, A. Yasin, S. M. Usman, S. Hussain, S. Khalid, and S. S. Ullah, "Exploring lightweight deep learning solution for malware detection in iot constraint environment," *Electronics*, vol. 11, no. 24, pp. 4147–4147, 2022.

[15] S. Anand, B. Mitra, S. Dey, A. Rao, R. Dhar, and J. Vaidya, "Malite: Lightweight malware detection and classification for constrained devices," *arXiv.org*, vol. abs/2309.03294, 2023.

[16] "Lightweight, effective detection and characterization of mobile malware families," *IEEE Transactions on Computers*, vol. 71, no. 11, pp. 2982–2995, 2022.

[17] T. Bilot, N. E. Madhoun, K. A. Agha, and A. Zouaoui, "A survey on malware detection with graph representation learning," 2024.

[18] H. H. Liu, L. Gong, X. Mo, G. Dong, and J. Yu, "Ltachecker: Lightweight android malware detection based on dalvik opcode sequences using attention temporal networks," *IEEE Internet of Things Journal*, pp. 1–1, 2024.

[19] M. Krzyszton, B. Bok, M. Lew, and A. Sikora, "Lightweight on-device detection of android malware based on the koodous platform and machine learning," *Sensors*, vol. 22, no. 17, pp. 6562–6562, 2022.

[20] S. Laboratory, "Stratosphere laboratory iot dataset," 2025, accessed: 2025-01-13. [Online]. Available: https://www.stratosphereips.org/datasets-iot

[21] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6097–6105. [Online]. Available: https://arxiv.org/abs/1704.04861

[22] N. Sevani, K. Azizah, and W. Jatmiko, "A feature-based transfer learning to improve the image classification with support vector machine," *International Journal of Advanced Computer Science and Applications*, vol. 14, no. 6, 2023. [Online]. Available: http://dx.doi.org/10.14569/IJACSA.2023.0140632

[23] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6100–6108. [Online]. Available: https://arxiv.org/abs/1704.04861

[24] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6105–6114. [Online]. Available: https://arxiv.org/abs/1905.11946

[25] X. Zhang, X. Li, and D. Xu, "Shufflenet: An extremely efficient convolutional neural network for mobile devices," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 6848–6856. [Online]. Available: https://arxiv.org/abs/1707.01083

[26] F. J. Iandola, M. Moskewicz, K. Ashraf, W. Dally, and K. Keutzer, "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and ¡0.5mb model size," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2016, pp. 2440–2448. [Online]. Available: https://arxiv.org/abs/1602.07360