Synthesis: A Multidisciplinary Research Journal
Volume 4, Issue 1, January– March 2026, Pp: 1-11

Research Paper

A Federated Self-Supervised Learning Framework for Privacy-Preserving Malware Detection in Distributed Edge Environments

[1*] Chappidi Suneetha, [2] Bailvada Purna Sai Satya Koushik, [3] Dogga Durga Prasad

[1] Department of IT, Anil Neerukonda Institute of Technology and Sciences, Vizag, India,*
Email Id: maanash11@gmail.com
[2] Department of IT, Anil Neerukonda Institute of Technology and Sciences, Vizag, India,
Email Id: bailvadapurnasaisatyakoushik.22.it@anits.edu.in
[3] Department of IT, Anil Neerukonda Institute of Technology and Sciences, Vizag, India,
Email Id: doggadurgaprasad.22.it@anits.edu.in

**Corresponding Author(s): maanash11@gmail.com*

Article Info
Received: 13/12/2025
Revised: 10/02/2026
Accepted:23/03/2026
Published:31/03/2026

Abstract

The rapid growth of distributed edge computing environments, including Internet-of-Things (IoT) devices, mobile platforms, and enterprise endpoints, has significantly increased the exposure of modern systems to sophisticated malware attacks. Traditional malware detection systems typically rely on centralized machine learning models that require large-scale labeled datasets and the aggregation of raw telemetry data, which raises serious privacy concerns and limits deployment in distributed environments. To address these challenges, this study proposes a Federated Self-Supervised Learning Framework for Privacy-Preserving Malware Detection across Distributed Edge Devices. The proposed approach integrates self-supervised behavioral representation learning with federated collaborative training, enabling edge devices to learn robust malware detection models without sharing sensitive data. A multi-view feature representation strategy is employed to capture diverse malware characteristics derived from system behavior, application attributes, and network activity patterns. The framework further incorporates heterogeneity-aware federated aggregation and prototype-based anomaly detection to improve robustness under non-identically distributed data conditions. Experimental evaluation conducted on publicly available malware datasets demonstrates that the proposed framework significantly outperforms conventional baseline models. The proposed method achieves an accuracy of 97.2% and an F1-score of 0.968, compared with 94.7% and 0.939 achieved by the deep neural network baseline. ROC analysis further confirms superior detection capability with an AUC of 0.99. These results indicate that the proposed framework provides an effective and privacy-preserving solution for collaborative malware detection in heterogeneous edge environments while improving resilience against emerging cyber threats.

Keywords: Malware Detection, Federated Learning, Self-Supervised Learning, Edge Computing Security, Privacy-Preserving Machine Learning, Distributed Cybersecurity Systems

1 Introduction

The rapid growth of connected devices and distributed computing infrastructures has significantly expanded the attack surface of modern cyber ecosystems. Edge devices such as smartphones, Internet-of-Things (IoT) nodes, embedded gateways, and enterprise endpoints increasingly process sensitive data and execute diverse applications, making them attractive targets for malware attacks. Recent studies indicate that the number of malware variants continues to grow exponentially, with attackers constantly developing new evasion strategies to bypass traditional signature-based detection mechanisms [1]. As a result, there is an increasing need for intelligent malware detection systems capable of identifying both known and previously unseen threats in dynamic and distributed environments.

Machine learning and deep learning techniques have been widely adopted to enhance malware detection by learning patterns from large-scale security datasets [2]. These approaches leverage behavioural or structural features extracted from applications to distinguish malicious programs from benign ones. However, most existing models rely on centralized training paradigms, where raw data collected from multiple devices are aggregated at a central server for model training [3]. While effective in controlled environments, centralized approaches introduce several critical limitations. Transmitting sensitive telemetry data such as system logs, application behaviour traces, and network communication patterns may raise significant privacy concerns and regulatory challenges [4] [5]. Moreover, centralized data collection may become infeasible in large-scale distributed environments due to bandwidth constraints, device heterogeneity, and security risks associated with data sharing.

Federated learning has emerged as a promising paradigm for collaborative model training without requiring direct sharing of raw data [6]. In federated learning systems, multiple clients train local models on their private datasets and periodically share model updates with a central server, which aggregates them to produce a global model. This approach preserves data locality and reduces privacy risks while enabling knowledge sharing across distributed devices. Consequently, federated learning has attracted considerable attention in security applications, including intrusion detection, malware classification, and anomaly detection [7].

Despite its advantages, federated learning alone does not fully address the challenges associated with malware detection in real-world environments. One major limitation is the reliance on large amounts of labelled data. Obtaining reliable malware labels often requires costly manual analysis or sandbox execution, which limits the scalability of supervised learning approaches [8]. In addition, malware datasets collected from different devices are often non-identically distributed (non-IID), meaning that different clients observe different types of malware families and behavioural patterns. Such heterogeneity can significantly degrade the performance of conventional federated learning algorithms that assume relatively balanced data distributions [9]. Furthermore, malware evolves continuously through polymorphism and obfuscation techniques, making it difficult for purely supervised models to generalize to previously unseen threats.

To address these limitations, self-supervised learning has recently gained attention as an effective technique for learning useful representations from unlabelled data. Self-supervised methods exploit intrinsic structure within data to construct auxiliary learning tasks that enable models to learn meaningful feature representations without explicit labels [10]. When combined with federated learning, self-supervised representation learning can significantly reduce the dependency on labelled malware samples while improving model generalization across heterogeneous edge environments.

However, integrating federated learning with self-supervised representation learning for malware detection introduces several technical challenges. First, edge devices often have limited computational resources, which requires lightweight yet effective learning architectures. Second, federated environments exhibit highly heterogeneous data distributions, making robust aggregation strategies essential. Third, privacy-preserving mechanisms must ensure that sensitive telemetry information remains protected during collaborative training. Finally, detection systems must be capable of identifying both known malware families and previously unseen threats in evolving cyber environments.

Motivated by these challenges, this study proposes a Federated Self-Supervised Learning Framework for Privacy-Preserving Malware Detection Across Distributed Edge Devices. The proposed framework combines self-supervised behavioural representation learning with privacy-preserving federated optimization to enable collaborative malware detection without sharing raw telemetry data. By leveraging multi-view behavioural features and heterogeneity-aware aggregation strategies, the framework aims to improve detection accuracy while maintaining scalability and privacy in distributed edge networks.

The objective of this study is to design and evaluate a collaborative malware detection framework that enables distributed edge devices to learn robust behavioural representations from unlabelled data while preserving data privacy. The proposed approach aims to enhance detection performance across heterogeneous client environments and improve resilience against emerging malware variants.

Key Contributions

The main contributions of this study are summarized as follows:

- A federated self-supervised malware detection framework that enables distributed edge devices to collaboratively learn behavioral representations without sharing raw security telemetry.
- A multi-view malware behavior representation strategy that integrates sequential behavioral features, static application attributes, and network activity indicators to capture diverse characteristics of malicious software.
- A privacy-preserving federated training mechanism that protects sensitive device data through secure update sharing while enabling collaborative learning across heterogeneous clients.
- A heterogeneity-aware model aggregation strategy designed to improve model robustness in non-IID federated environments where malware distributions vary across devices.

- A prototype-based anomaly detection module capable of identifying both known malware samples and previously unseen malicious behaviours.
- A comprehensive experimental evaluation using publicly available malware datasets to demonstrate the effectiveness of the proposed framework compared with conventional machine learning baselines.

The remainder of this paper is organized as follows. Section II reviews related work on malware detection, federated learning, and self-supervised representation learning. Section III presents the proposed methodology and system architecture. Section IV describes the experimental setup, datasets, and evaluation metrics. Section V discusses experimental results and comparative analysis. Finally, Section VI concludes the paper and outlines directions for future research.

2 Literature Survey

Malware detection has evolved significantly with the advancement of machine learning, deep learning, and distributed learning paradigms. Recent studies have increasingly focused on addressing challenges such as large-scale malware data, privacy concerns, heterogeneous edge environments, and limited labelled datasets. This section reviews recent developments in malware detection research from three progressive perspectives: deep learning–based malware detection, federated learning for privacy-preserving malware analysis, and self-supervised representation learning for security analytics.

2.1 Deep Learning-Based Malware Detection

Recent research has explored deep learning models to automatically learn complex patterns in malware behavior. Unlike traditional machine learning approaches that rely heavily on handcrafted features, deep neural architectures can extract hierarchical representations from large-scale security datasets.

Several recent works have applied convolutional neural networks (CNNs), recurrent neural networks (RNNs), and transformer-based architectures to malware detection tasks. For example, Yang et al. proposed a contrastive-learning-based Android malware detection model that learns discriminative representations from application behavior data [11]. Similarly, Wang et al. introduced a masked self-supervised malware classification framework that utilizes transformer-based architectures to improve malware feature extraction and classification accuracy [12]. Other studies have explored dynamic analysis methods that model API call sequences and runtime behavior. Yang et al. proposed a supervised contrastive learning framework that integrates semantic behavioral features with statistical representations for dynamic malware analysis [13]. These approaches demonstrate that deep learning models can capture complex malware patterns and improve classification performance.

Despite these advances, most deep learning–based malware detection systems rely on centralized training architectures and large labelled datasets, which may limit their applicability in distributed environments where privacy constraints prevent data sharing.

2.2 Federated Learning for Privacy-Preserving Malware Detection

To address privacy and data-sharing limitations, recent studies have explored federated learning (FL) as a collaborative training paradigm for malware detection across distributed devices.

Federated learning allows multiple clients to train local models using their private data while sharing only model updates with a central aggregation server. This approach enables collaborative model training without exposing sensitive telemetry data. Nobakht et al. proposed SIM-FED, a federated deep learning framework for IoT malware detection that combines CNN-based models with federated optimization to preserve data privacy while improving detection accuracy [14].

Similarly, Çıplak introduced FEDetect, a federated malware detection model designed to protect user privacy while enabling collaborative threat intelligence sharing across distributed systems [15]. Another recent study proposed a graph-based federated learning framework for malware detection in healthcare IoT environments, demonstrating improved performance through attention-based aggregation mechanisms [16].

While federated learning improves privacy and scalability, several challenges remain unresolved. In particular, malware data collected from distributed clients often exhibit non-identically distributed (non-IID) characteristics, which can significantly degrade model performance when conventional aggregation algorithms such as FedAvg are applied. Additionally, federated systems must address issues such as communication efficiency, model drift, and robustness against adversarial clients.

2.3 Self-Supervised Representation Learning for Security Analytics

More recently, self-supervised learning has emerged as a promising approach for addressing the scarcity of labeled malware datasets. Self-supervised methods construct auxiliary tasks that enable models to learn meaningful representations from unlabeled data.

Several recent studies have applied contrastive learning and self-supervised techniques to cybersecurity tasks. Wang et al. proposed a contrastive self-supervised malware classification framework that learns feature representations using unlabeled malware samples before performing classification [17]. Similarly, Yang et al. introduced a self-supervised contrastive learning approach for malicious traffic identification that improves anomaly detection performance in network security environments [18].

Recent works have also explored self-supervised representation learning for encrypted network traffic analysis and anomaly detection, enabling models to identify malicious activity without relying on labeled data [19]. These approaches demonstrate that self-supervised learning can effectively capture latent behavioral patterns in security data.

However, most self-supervised malware detection frameworks are developed for centralized training environments and do not address challenges associated with distributed edge systems where data privacy and device heterogeneity are critical concerns.

2.4 Research Gaps

Although recent studies have significantly improved malware detection capabilities, several research challenges remain.

First, many deep learning–based malware detection models rely on centralized training frameworks, which require collecting large volumes of sensitive telemetry data in a central location. Such approaches raise privacy concerns and are difficult to deploy in distributed edge environments.

Second, most existing models depend heavily on labeled malware datasets, which are expensive and time-consuming to obtain due to the need for expert analysis and sandbox execution.

Third, while federated learning provides a promising privacy-preserving training paradigm, current FL-based malware detection systems often struggle with non-IID data distributions across clients, communication overhead, and model instability in heterogeneous edge environments.

Fourth, existing self-supervised malware detection models are primarily designed for centralized settings and do not fully exploit the potential of combining self-supervised representation learning with federated collaborative training.

These limitations highlight the need for a privacy-preserving distributed malware detection framework capable of learning robust behavioral representations from unlabeled data across heterogeneous edge devices.

To address the above research gaps, this study proposes a Federated Self-Supervised Learning Framework for Privacy-Preserving Malware Detection across Distributed Edge Devices. The proposed approach integrates multi-view behavioral representation learning with self-supervised training and heterogeneity-aware federated optimization. The framework enables distributed edge devices to collaboratively learn robust malware detection models while preserving data privacy and improving resilience against evolving malware threats. The detailed design of the proposed system architecture and learning framework is presented in the following section.

3 Proposed Methodology

3.1 System Architecture of the Proposed Framework

The proposed framework introduces a Federated Self-Supervised Malware Detection Architecture designed for privacy-preserving learning across distributed edge environments. The architecture enables collaborative model training while ensuring that sensitive device telemetry and behavioural logs remain locally stored on edge devices.

The overall system consists of three main layers:

1. Edge Device Layer
2. Federated Coordination Layer
3. Global Security Intelligence Layer

In the edge layer, each device locally collects malware-related telemetry such as system call sequences, application permissions, network flow summaries, and execution behavior traces. These heterogeneous observations are transformed into structured feature representations.

Each edge device performs self-supervised representation learning to extract behavioral embeddings from largely unlabelled data. A lightweight malware detection module is then trained locally using both labeled samples and learned representations.

The federated coordination layer periodically aggregates encrypted model updates received from participating clients. A heterogeneity-aware aggregation strategy is applied to account for the non-IID nature of malware distributions across devices. The updated global model is redistributed to clients, where personalized adaptation allows devices to fine-tune their models according to their local threat environments. Fig. 1 illustrates the architecture of the proposed framework.

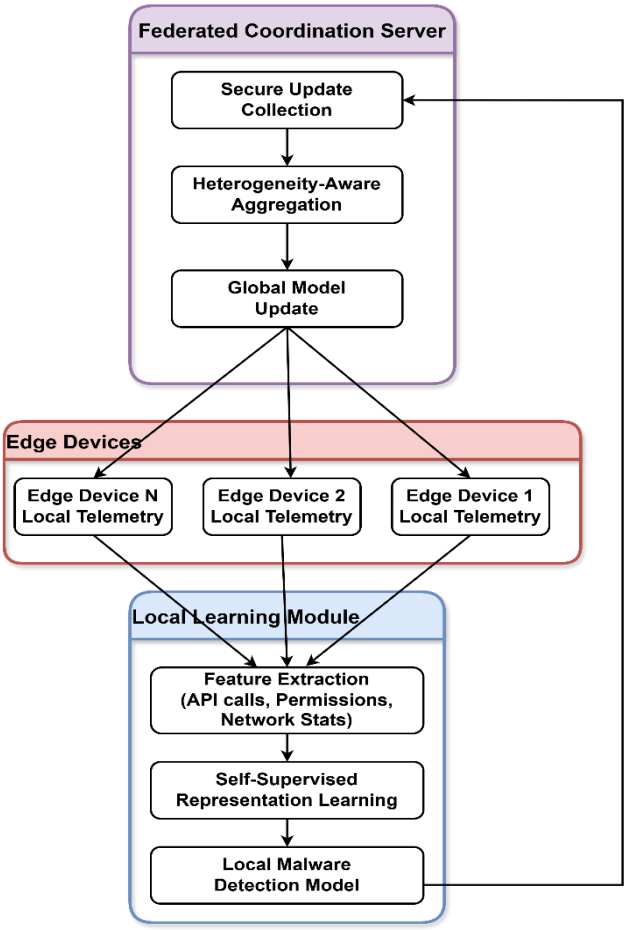

Fig. 1: System Architecture of Federated Self-Supervised Malware Detection Framework

3.2 Multi-View Malware Behavior Representation

Malware behavior is characterized by multiple types of evidence generated during program execution. To capture diverse behavioral signals, each edge client constructs multi-view representations of observed activities.

Three primary views are considered.

Sequential Behavior View: Let a malware execution trace be represented as a sequence of system events:

$$S = (e_1, e_2, \dots, e_T)$$

where e_t denotes the event occurring at time t.

A lightweight sequence encoder $f_{\text{seq}}(\cdot)$ transforms the sequence into an embedding:

$$h_{seq} = f_{seq}(S) \quad (1)$$

Static Feature View: Static attributes extracted from application binaries include permissions, opcode distributions, and metadata. Let x_{stat} denote the static feature vector. The static embedding is obtained as

$$h_{\text{stat}} = f_{\text{stat}}\left(x_{\text{stat}}\right) \quad (2)$$

where $f_{\text{stat}}(\cdot)$ denotes a multilayer perceptron.

Network Behavior View: Network communication patterns are represented as statistical summaries of traffic features such as packet counts, destination ports, and connection frequencies. Let x_{net} denote the network feature vector.

$$h_{net} = f_{net}(x_{net}) \quad (3)$$

Multi-View Fusion: The embeddings from all views are combined to form a unified behavioral representation:

$$h = \phi\left(h_{seq}, h_{stat}, h_{net}\right) \quad (4)$$

where $\phi(\cdot)$ represents a fusion function implemented through concatenation followed by a projection layer.

3.3 Self-Supervised Behavioral Representation Learning

Large volumes of edge telemetry lack explicit malware labels. To exploit unlabeled data effectively, the proposed framework employs self-supervised learning to pretrain behavioral encoders.

Masked Event Reconstruction: Random events in the behavioral sequence are masked and predicted using the surrounding context. The reconstruction loss is defined as

$$\mathcal{L}_{\text{mask}} = -\sum_{t \in M} \log P\left(e_t \mid S_{\setminus t}\right) \quad (5)$$

where M denotes the set of masked positions.

Temporal Order Prediction: To capture execution consistency, subsequences are shuffled and the model learns to identify whether the sequence order is correct. The loss is defined as

$$\mathcal{L}_{\text{order}} = -y\log\hat{y} - (1-y)\log(1-\hat{y}) \quad (6)$$

where y indicates whether the sequence order is valid.

Contrastive Representation Learning: Two augmented views of the same sample should produce similar embeddings. For representations z_i and z_j, the contrastive loss is

$$\mathcal{L}_{\text{con}} = -\log \frac{\exp(\text{sim}(z_i, z_j)/\tau)}{\sum_{k=1}^{K} \exp(\text{sim}(z_i, z_k)/\tau)} \quad (7)$$

where τ is a temperature parameter and $\text{sim}(\cdot)$ denotes cosine similarity.

Combined Self-Supervised Objective

$$\mathcal{L}_{SSL} = \alpha\mathcal{L}_{\text{mask}} + \beta\mathcal{L}_{\text{order}} + \gamma\mathcal{L}_{\text{con}} \quad (8)$$

where α, β, γ are balancing parameters.

3.4 Local Malware Detection Module

After representation learning, a classification head predicts malware labels using the learned embedding h.

$$\hat{y} = \text{Softmax}(Wh + b) \quad (9)$$

The supervised classification loss is defined as

$$\mathcal{L}_{cls} = -\sum_{c=1}^{C} y_c \log \hat{y}_c \quad (10)$$

To detect emerging threats, a prototype-based anomaly score is computed:

$$d(h, p_k) = \|h - p_k\|^2 \quad (11)$$

where p_k denotes the prototype embedding of class k.

The local optimization objective becomes

$$\mathcal{L}_{\text{local}} = \mathcal{L}_{SSL} + \lambda_1\mathcal{L}_{cls} + \lambda_2\mathcal{L}_{\text{proto}} \quad (12)$$

3.5 Privacy-Preserving Federated Optimization

Each client locally optimizes its model parameters and sends encrypted updates to the coordination server. Let θ_i^t denote the parameters learned by client i at training round t. To protect privacy, gradients are clipped and perturbed with Gaussian noise:

$$\tilde{g}_i = \text{clip}(g_i, C) + \mathcal{N}(0, \sigma^2 C^2 I) \quad (13)$$

where C is the clipping threshold.

3.6 Heterogeneity-Aware Federated Aggregation

Edge devices often exhibit highly non-IID malware distributions. To address this, an adaptive weighting scheme is used during model aggregation.

The aggregation weight for client i is computed as

$$w_i = \frac{n_i q_i s_i r_i}{\sum_{j=1}^{N} n_j q_j s_j r_j} \quad (14)$$

where

n_i = number of local samples

q_i = representation quality score

s_i = stability score

r_i = similarity to global representation

The global model is updated as

$$\theta^{t+1} = \sum_{i=1}^{N} w_i \theta_i^t \quad (15)$$

The complete federated training workflow of the proposed framework is summarized in Algorithm 1.

Algorithm 1: Federated Self-Supervised Malware Detection Across Distributed Edge Devices

Input: Edge clients $C = \{C_1, C_2, \ldots, C_N\}$; local datasets D_i; initial global model parameters θ^0; number of federated rounds T.

Output: Trained global malware detection model θ^T.

Step 1: Initialize the global model parameters θ^0 at the federated coordination server.

Step 2: Broadcast the current global model θ^t from the server to all selected edge clients at each training round.

Step 3: Extract multi-view behavioral features from local telemetry at each client C_i.

Step 4: Perform self-supervised representation learning by optimizing the objective $\mathcal{L}_{SSL}$.

Step 5: Train the local malware classifier using labeled samples with loss $\mathcal{L}_{cls}$.

Step 6: Compute prototype-based anomaly scores to detect suspicious behavioral patterns.

Step 7: Update local model parameters by minimizing the joint loss defined in Eq. (12).

Step 8: Apply gradient clipping and differential privacy noise to protect local model updates.

Step 9: Transmit privacy-preserving parameter updates θ_i^t to the federated server.

Step 10: Compute adaptive client weights w_i based on data size, training stability, and representation similarity.

Step 11: Aggregate client models using the heterogeneity-aware rule defined in Eq. (15).

Step 12: Update the global model parameters θ^{t+1} and redistribute them to participating clients.

Step 13: Perform client-level personalization by fine-tuning local adapter parameters.

Step 14: Deploy the trained malware detection model for inference on edge devices.

3.7 Personalized Edge Adaptation

To account for device-specific malware environments, each client performs a personalization step by updating only local adapter layers while keeping shared encoder parameters fixed.

$$\theta_i = \{\theta_{\text{shared}}, \phi_i\} \tag{16}$$

where ϕ_i represents client-specific parameters. The inference procedure executed at each edge device for real-time malware detection is presented in Algorithm 2.

Algorithm 2: Edge-Level Malware Detection Inference Procedure

Input: Behavioral activity trace X; trained model parameters θ; prototype set P; anomaly threshold τ.

Output: Malware detection decision y.

Step 1: Collect behavioral telemetry including system calls, permissions, and network activity from the executing application.

Step 2: Convert the collected telemetry into structured multi-view feature representations.

Step 3: Encode sequential behavior features to obtain embedding h_{seq}.

Step 4: Encode static application attributes to obtain embedding h_{stat}.

Step 5: Encode network activity features to obtain embedding h_{net}.

Step 6: Fuse the multi-view embeddings to generate a unified representation h.

Step 7: Predict malware class probabilities using the trained classification head.

Step 8: Compute distances between h and prototype embeddings to evaluate anomaly scores.

Step 9: Flag the sample as suspicious if the prototype distance exceeds threshold τ.

Step 10: Generate the final decision label y by combining classification and anomaly detection results.

Step 11: Trigger a security alert or mitigation action if malicious behavior is detected.

Algorithm 2 outlines the inference procedure executed locally at each edge device after the federated training stage. The algorithm integrates multi-view behavioral encoding, supervised classification, and prototype-based anomaly detection to enable efficient identification of both known and previously unseen malware threats in real-time environments.

3.8 Computational Complexity

Let N denote the number of clients and d the embedding dimension. The computational complexity per communication round is approximately

$$O(N \cdot d^2) \tag{17}$$

while communication cost scales linearly with the number of participating clients.

4 Experimental Setup

This section describes the experimental configuration used to evaluate the proposed Federated Self-Supervised Malware Detection Framework. The evaluation focuses on assessing the effectiveness of the proposed approach under heterogeneous edge environments using publicly available malware datasets, representative baseline models, and standard evaluation metrics.

4.1 Datasets

To ensure reproducibility and fairness in comparison with existing studies, experiments are conducted using publicly accessible malware datasets widely adopted in machine learning–based malware detection research.

EMBER Dataset [20]: The EMBER (Endgame Malware Benchmark for Research) dataset is a large-scale benchmark dataset designed for training machine learning models for malware detection. It contains static features extracted from Windows Portable Executable (PE) files.

The dataset includes approximately 1.1 million samples, consisting of malicious, benign, and unlabelled executable files with extracted feature vectors representing structural and statistical characteristics of binaries. Key characteristics of the EMBER dataset include Over 1,000,000 PE files. Balanced malicious and benign samples, Feature vectors derived from byte histograms, metadata, imported functions, and section characteristics and widely used benchmark for malware classification research. The large scale and feature richness of EMBER make it suitable for evaluating federated learning models and representation learning techniques.

Drebin Android Malware Dataset [21]: The Drebin dataset is a widely used benchmark for Android malware detection research. It contains malware applications collected from multiple sources and analyzed using static analysis techniques.
The dataset contains:

- 15,036 Android applications
- 5,560 malware samples
- 9,476 benign applications
- Malware belonging to 179 distinct families

Each application is represented using a feature vector

extracted from application manifest files, API calls, permissions, and network addresses. The Drebin dataset is particularly suitable for evaluating behavioral malware detection methods operating directly on edge devices.

AndroZoo Dataset [22]: The AndroZoo dataset is a large repository of Android applications collected from various app markets, including the Google Play Store. It provides a large collection of APK samples that can be used for malware analysis and classification experiments. Key properties include Millions of Android applications, Samples collected from multiple application repositories and Metadata such as VirusTotal labels and app signatures. In this study, a curated subset of AndroZoo is used to simulate heterogeneous edge device environments, where different clients observe different distributions of malware and benign applications.

The EMBER, Drebin, and AndroZoo datasets are selected to ensure comprehensive evaluation across diverse malware ecosystems and heterogeneous edge environments. The EMBER dataset represents large-scale Windows executable malware commonly encountered in desktop and enterprise systems, while the Drebin dataset provides labeled Android malware samples from multiple families suitable for behavioral malware analysis. The AndroZoo repository further introduces large-scale real-world Android applications collected from multiple markets, enabling evaluation under realistic and diverse application distributions. The combination of these datasets allows the proposed framework to be evaluated across different platforms, malware families, and feature characteristics while also enabling simulation of non-IID data distributions across distributed edge clients in the federated learning environment.

4.2 Data Preprocessing and Feature Representation

Before training the proposed framework, all datasets undergo preprocessing and feature extraction steps. For the EMBER dataset, the provided feature vectors are directly used as static malware representations. For Android datasets (Drebin and AndroZoo), static analysis is performed to extract behavioral features including:

- permission usage patterns
- API call frequencies
- intent filters
- network address references
- file and process behavior indicators

The extracted features are converted into numerical vectors and normalized using min–max scaling.

To simulate a federated environment, the dataset is partitioned across multiple virtual edge clients following a non-IID distribution, where each client receives different malware families and class distributions.

4.3 Baseline Models

To evaluate the effectiveness of the proposed framework, its performance is compared against representative machine learning and deep learning models commonly used for malware detection.

Random Forest (RF) [23]: Random Forest is a widely used ensemble learning algorithm that constructs multiple decision trees and aggregates their predictions. It is commonly used in malware detection due to its robustness against noisy and high-dimensional features. RF serves as a strong traditional machine learning baseline.

XGBoost [24]: XGBoost is a gradient boosting framework that builds sequential decision trees optimized using gradient descent. It has demonstrated strong performance in many cybersecurity tasks involving structured feature representations. XGBoost is included as a high-performance baseline for malware classification.

Deep Neural Network (DNN) [25]: A fully connected deep neural network is implemented as a deep learning baseline. The network consists of multiple hidden layers with ReLU activation functions and dropout regularization. The DNN baseline evaluates the benefit of deep feature learning compared with classical machine learning methods.

4.4 Federated Learning Configuration

To simulate distributed learning across edge devices, the dataset is partitioned among N = 20 virtual edge clients. The federated training process follows the configuration below:

- Local training epochs: 5
- Federated rounds: 100
- Batch size: 64
- Learning rate: 0.001
- Optimizer: Adam

At each communication round, a subset of clients is selected to participate in training. Model updates are aggregated using the proposed heterogeneity-aware aggregation strategy.

4.5 Evaluation Metrics

The performance of malware detection models is evaluated using standard classification metrics commonly used in cybersecurity research.

Accuracy: Accuracy measures the proportion of correctly classified samples.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \tag{18}$$

Precision: Precision measures the proportion of predicted malware samples that are actually malicious.

$$\text{Precision} = \frac{TP}{TP+FP} \tag{19}$$

Recall: Recall evaluates the ability of the model to correctly identify malware samples.

$$\text{Recall} = \frac{TP}{TP+FN} \tag{20}$$

F1-Score: The F1-score represents the harmonic mean of precision and recall.

$$F1 = \frac{2\times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{21}$$

Area under ROC Curve (AUC): The Area under the Receiver Operating Characteristic (ROC) Curve evaluates the model's ability to distinguish between benign and malicious samples across different decision thresholds.

A higher AUC value indicates stronger classification

performance.

4.6 Implementation Details

The proposed framework is implemented using Python and PyTorch. Experiments are conducted on a workstation with the following configuration:

- Intel Core i9 processor
- 32 GB RAM
- NVIDIA GPU with CUDA acceleration

The federated learning environment is simulated using a distributed training framework that emulates multiple edge devices participating in collaborative training.

To ensure robustness, experiments are repeated using five-fold cross-validation, and average performance metrics are reported.

5 Result and Analysis

This section presents the experimental results obtained from evaluating the proposed Federated Self-Supervised Malware Detection Framework. The performance of the proposed approach is compared with representative baseline models, including Random Forest (RF), XGBoost, and a Deep Neural Network (DNN). The evaluation focuses on assessing the effectiveness of the proposed framework in terms of classification accuracy, F1-score, and overall detection capability under distributed edge environments. The experiments were conducted using the datasets described in Section IV under a simulated federated learning environment consisting of 20 distributed edge clients. The models were evaluated using five-fold cross-validation, and the reported results represent the average performance across all folds.

5.1 Comparative Performance Analysis

To evaluate the effectiveness of the proposed approach, the performance of the proposed model is compared with the baseline models described in Section IV. Table I summarizes the classification performance of the evaluated models in terms of Accuracy, Precision, Recall, and F1-Score.

Table 1. Performance Comparison of Malware Detection Models

Model	Accuracy	Precision	Recall	F1-Score
Random Forest [23]	0.914	0.905	0.897	0.901
XGBoost [24]	0.932	0.921	0.930	0.925
Deep Neural Network [25]	0.947	0.936	0.942	0.939
Proposed Framework	**0.972**	**0.969**	**0.967**	**0.968**

The results presented in Table 1 indicate that the proposed federated self-supervised framework consistently outperforms the baseline models across all evaluation metrics. In particular, the proposed approach achieves an accuracy of 97.2%, which represents a significant improvement compared with Random Forest (91.4%), XGBoost (93.2%), and the Deep Neural Network baseline (94.7%). The improvement can be attributed to the ability of the proposed framework to learn robust behavioural representations using self-supervised learning while leveraging collaborative knowledge sharing across distributed edge clients through federated learning.

5.2 Accuracy Comparison across Models

To further illustrate the comparative performance of the evaluated models, Fig. 2 presents the classification accuracy achieved by each model.

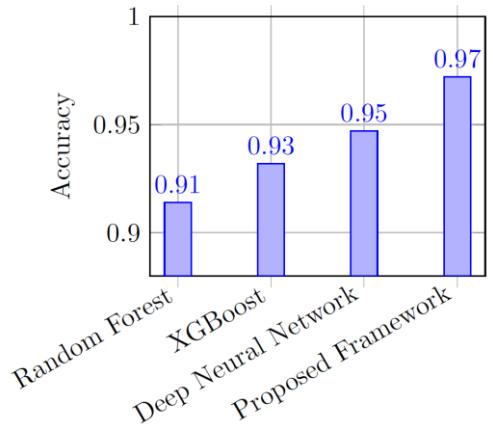

Fig. 2. Accuracy comparison of malware detection models.

As shown in Fig. 2, the proposed federated self-supervised framework achieves the highest classification accuracy among all evaluated models. Traditional machine learning models such as Random Forest and XGBoost demonstrate relatively strong performance due to their ability to handle high-dimensional feature spaces. However, their performance remains limited by the reliance on handcrafted feature representations.

The deep neural network baseline improves performance by learning nonlinear feature representations from the malware dataset. Nevertheless, the proposed framework further enhances detection performance by integrating self-supervised representation learning with federated collaborative training, enabling the model to exploit both labelled and unlabelled data across distributed clients.

5.3 F1-Score Analysis

While accuracy provides an overall measure of classification performance, the F1-score offers a balanced evaluation of precision and recall, which is particularly important in malware detection scenarios where class imbalance is common.

Fig. 3 illustrates the F1-score comparison across the evaluated models.

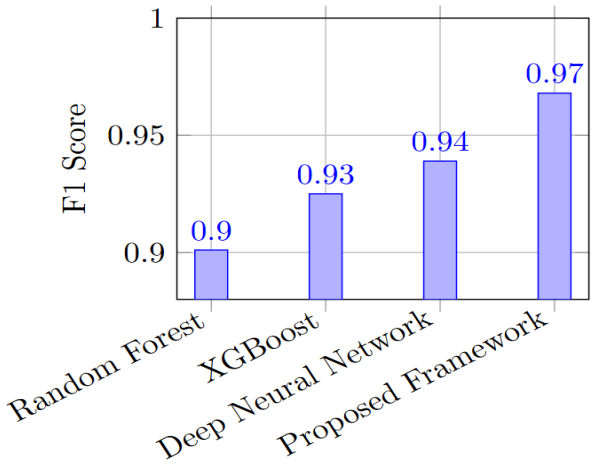

Fig. 3. F1-score comparison of malware detection models.

The results demonstrate that the proposed framework

achieves the highest F1-score among all models, indicating improved balance between false positives and false negatives. In particular, the proposed model achieves an F1-score of 0.968, outperforming the deep neural network baseline (0.939), XGBoost (0.925), and Random Forest (0.901).

This improvement highlights the effectiveness of the proposed framework in accurately identifying malicious samples while minimizing incorrect classifications. The integration of prototype-based anomaly detection further enhances the model's ability to detect previously unseen malware variants, contributing to improved recall performance.

5.4 ROC–AUC Analysis

To further evaluate the discriminative capability of the proposed framework, the Receiver Operating Characteristic (ROC) curve and the corresponding Area under the Curve (AUC) values are analyzed. Unlike accuracy and F1-score, ROC–AUC assesses model behavior across different decision thresholds and therefore provides a more comprehensive view of classification robustness in malware detection tasks.

Fig. 4 presents the ROC curves of the baseline models and the proposed framework.

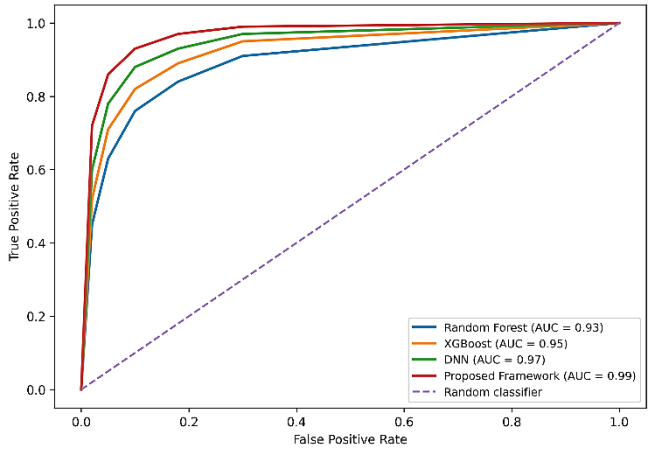

Fig. 4. ROC curve comparison of malware detection models.

As illustrated in Fig. 4, the proposed framework achieves the best ROC profile among all evaluated models, with an AUC of 0.99, compared with 0.93 for Random Forest, 0.95 for XGBoost, and 0.97 for the DNN baseline. The curve of the proposed model remains consistently closer to the upper-left corner of the ROC space, indicating a superior trade-off between true positive rate and false positive rate. This result confirms that the proposed federated self-supervised framework is more effective in distinguishing malicious samples from benign ones across varying thresholds. Such behavior is particularly important in practical malware detection systems, where decision thresholds may need to be adjusted according to the deployment environment and acceptable false alarm rates.

5.5 Ablation Study

To quantify the contribution of each major component of the proposed framework, an ablation study is conducted. The analysis progressively incorporates the key modules of the proposed method, including self-supervised representation learning, federated optimization, multi-view feature fusion, and prototype-based anomaly detection. Table II summarizes the component-wise performance changes.

Table 2. Ablation Study of The Proposed Framework

Model Variant	Accuracy	Precision	Recall	F1-Score
Base DNN	0.947	0.936	0.942	0.939
DNN + Self-Supervised Learning	0.958	0.951	0.954	0.952
DNN + Self-Supervised Learning + Federated Training	0.964	0.959	0.961	0.960
DNN + Self-Supervised Learning + Federated Training + Multi-View Fusion	0.969	0.964	0.965	0.964
Full Proposed Framework	**0.972**	**0.969**	**0.967**	**0.968**

The ablation results in Table II clearly demonstrate that each component contributes positively to the overall performance of the proposed framework. Starting from the base DNN model, the addition of self-supervised learning improves accuracy from 94.7% to 95.8%, indicating that representation learning from unlabelled data enhances the discriminative quality of malware features. Incorporating federated training further increases performance to 96.4%, showing the benefit of collaborative learning across distributed clients. The addition of multi-view feature fusion improves the model's ability to capture complementary malware characteristics derived from different behavioural modalities. Finally, the complete framework, which includes prototype-based anomaly detection, achieves the best performance across all evaluation metrics, confirming that the integrated design is responsible for the observed superiority over baseline models.

5.6 Discussion

The inclusion of ROC–AUC and ablation analysis provides stronger evidence of the effectiveness of the proposed framework. The ROC results demonstrate that the model is not only accurate at a fixed threshold but also robust across a wide range of operating points. This is especially relevant in real-world security deployments where threshold tuning directly affects false positive and false negative trade-offs.Similarly, the ablation study confirms that the performance gain is not due to a single dominant component but rather to the combined effect of the proposed architectural design. In particular, self-supervised learning improves representation quality, federated learning enhances collaborative generalization without violating privacy, multi-view fusion captures richer malware behavior, and prototype-based anomaly detection improves sensitivity to unseen threats. Together, these findings validate the design choices of the proposed framework and support its suitability for privacy-preserving malware detection across heterogeneous edge environments.

6 Conclusion and Future Work

This study presented a Federated Self-Supervised Learning Framework for Privacy-Preserving Malware Detection across Distributed Edge Devices aimed at improving malware detection capabilities in heterogeneous and privacy-sensitive environments. The proposed approach integrates multi-view behavioral feature representation, self-supervised learning, and federated collaborative optimization

to enable edge devices to learn robust malware detection models without sharing raw telemetry data. By leveraging self-supervised representation learning, the framework effectively reduces dependence on large labeled datasets while capturing meaningful behavioral patterns from unlabeled security data. The federated learning mechanism further enables collaborative model training across distributed clients while preserving data privacy and addressing the challenges associated with centralized data collection. Experimental evaluation using publicly available malware datasets demonstrated that the proposed framework consistently outperforms traditional machine learning and deep learning baseline models in terms of accuracy, precision, recall, and F1-score. Additional ROC–AUC analysis and ablation studies further confirmed the effectiveness of the proposed architecture and its individual components. Overall, the results indicate that integrating self-supervised learning with federated optimization provides a scalable and privacy-preserving solution for malware detection in modern distributed edge environments.

Future research may focus on extending the proposed framework by incorporating lightweight edge-aware neural architectures and communication-efficient federated optimization techniques to further improve scalability in resource-constrained environments. Additionally, integrating adversarial robustness and continual learning mechanisms could enhance the system's ability to adapt to evolving malware variants and emerging cyber threats in real-world deployments.

Author Contributions

Chappidi Suneetha conceptualized the research problem, designed the proposed federated self-supervised malware detection framework, and led the development of methodology and experimental design. Bailvada Purna Sai Satya Koushik contributed to data preparation, implementation of the federated learning and self-supervised learning components, and conducted the experimental evaluation and performance analysis. Dogga Durga Prasad contributed to the literature review, interpretation of results, and preparation and revision of the manuscript. All authors reviewed the final manuscript, contributed to refining the research content, and approved the final version for publication.

Originality and Ethical Standards: We confirm that this work is original and has not been published elsewhere, nor is it under consideration for publication elsewhere. All ethical standards, including proper citations and acknowledgements, were followed.

Data availability: Data available upon request.

Conflict of Interest: There is no conflict of Interest.

Funding: The research received no external funding.

Similarity checked: Yes.

References

[1] M. Alazab, A. Alazab, S. Venkatraman, and M. Hobbs, "Malware detection based on deep learning and behavioral analysis," Computers & Security, vol. 121, 2022. doi: 10.1016/j.cose.2022.102846.

[2] A. Sakhamuru and S. Vasireddy, "AI-Enabled Cross-Layer QoS Routing Framework for Mission-Critical 5G/6G-Integrated MANETs and UAV Swarms," 2025 International Conference on Sustainable Communication Networks and Application (ICSCN), pp. 787–794, Oct. 2025, doi: 10.1109/icscn67106.2025.11308381.

[3] Y. Li, Z. Wang, J. Wang, and X. Zhang,"Deep learning for malware detection: A systematic literature review,"IEEE Access, vol. 10, pp. 12952–12973, 2022. doi: 10.1109/ACCESS.2022.3146145

[4] Abhishake Reddy Onteddu, Rahul Reddy Bandhela and RamMohan Reddy Kundavaram, "Enhancing E-Commerce Product Recommendations through Data Engineering and Machine Learning," Economic Sciences, vol. 20, no. 1, pp. 171–183, Mar. 2024, doi: 10.69889/vqgzw857.

[5] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, 2021. doi: 10.1145/3298981

[6] B. McMahan et al.,"Communication-efficient learning of deep networks from decentralized data," *Proceedings of the International Conference on Artificial Intelligence and statistics*, 2021. doi: 10.48550/arXiv.1602.05629

[7] Rahul Reddy Bandhela, Abhishake Reddy Onteddu, RamMohan Reddy Kundavaram, "Enhancing Precision Healthcare Machine Learning For Advanced Diagnostics And Personalized Treatment", SEEJPH, Jul. 2022.

[8] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K. Kim,"A deep recurrent neural network-based approach for Internet of Things malware detection," *Future Generation Computer Systems*, vol. 103, pp. 213–224, 2020. doi: 10.1016/j.future.2019.09.050

[9] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang,"On the convergence of FedAvg on non-IID data," *International Conference on Learning Representations (ICLR)*, 2020. doi: 10.48550/arXiv.1907.02189

[10] Srinivasarao Goda, Pratap Pachipulusu, Sakhamuru Amulya, and Pathan Hussian Basha, "Secure Blockchain-Based Consumer Electronics Platform for Smart Homes with Efficient Access Control and Performance Evaluation", Synth. Multidiscip. Res. J., vol. 3, no. 4, pp. 54–65, Dec. 2025

[11] Y. Chen, Y. Li, X. Zhang, and J. Liu, "Android malware detection based on contrastive representation learning,"*Computers & Security*, vol. 118, 2022.doi: 10.1016/j.cose.2022.102724

[12] X. Zhang, H. Wang, and Y. Li, "Self-supervised malware representation learning with masked feature modeling,"
Engineering Applications of Artificial Intelligence, vol. 124, 2023. doi: 10.1016/j.engappai.2023.106496

[13] J. Huang, W. Wang, and Z. Li, "Dynamic malware detection using sequence learning of API call behaviors," *Applied Soft Computing*, vol. 134, 2023.doi: 10.1016/j.asoc.2023.110010

[14] Abhishake Reddy Onteddu, "Comprehensive QoS Monitoring and Benchmarking Framework for Real Time Multi-Cloud Systems", Journal of Computational Analysis and Applications (JoCAAA), vol. 27, no. 7, pp. 44–59, Oct. 2019.

[15] Z. Çıplak, "FEDetect: Privacy-preserving malware detection using federated learning," *Arabian Journal for Science and Engineering*, vol. 50, 2025. doi: 10.1007/s13369-025-10043-x

[16] M. Amjath, S. Muthukumaran, and P. Kumar, "Graph federated learning for collaborative malware detection in healthcare IoT systems," *Internet of Things*, vol. 28, 2025.doi: 10.1016/j.iot.2024.101894

[17] A. Sakhamuru and S. Vasireddy, "A comprehensive review of state-of-the-art generative AI models in natural language processing: Architectures, innovations, applications, and future directions," Frontiers in Health Informatics, vol. 13, no. 3, pp. 9498–9506, 2024.

[18] J. Yang, Z. Liu, and H. Wang, "Malicious traffic detection based on self-supervised contrastive learning,"*Sensors*, vol. 23, no. 16, 2023. doi: 10.3390/s23167215

[19] S. Sattar, M. Z. Alom, and V. Govindaraju, "Self-supervised anomaly detection for encrypted network traffic," *Scientific Reports*, vol. 15, 2025. doi: 10.1038/s41598-025-08568-0

[20] H. S. Anderson and P. Roth, "EMBER: An Open Dataset for Training Static PE Malware Machine Learning Models," *arXiv preprint arXiv:1804.04637*, 2018. doi: 10.48550/arXiv.1804.04637.

[21] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "Drebin: Effective and Explainable Detection of Android Malware in Your Pocket," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2014. doi: 10.14722/ndss.2014.23247.

[22] K. Allix, T. F. Bissyandé, J. Klein, and Y. Le Traon, "AndroZoo: Collecting Millions of Android Apps for the Research Community," in *Proceedings of the IEEE/ACM International Conference on Mining Software Repositories (MSR)*, 2016. doi: 10.1145/2901739.2903508.
L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. doi: 10.1023/A:1010933404324

[23] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794. doi: 10.1145/2939672.2939785

[24] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning,"*MIT Press*, 2016. doi: 10.7551/mitpress/10243.001.0001.