

Federated Transformer-DNN for Botnet Detection in Edge-IoT

Jyotsna A.¹, Mary Anita E. A.²

¹Department of Computer Science and Engineering, Christ University, Bangalore, Karnataka, India

Department of Artificial Intelligence and Machine Learning, Marian Engineering College (MEC), Thiruvananthapuram, Kerala, India

²Christ University, Bangalore, Karnataka, India

Abstract

The rapid growth of the Internet of Things (IoT) has introduced significant security challenges, particularly due to the rise in botnet attacks targeting distributed networks. This study investigates Intrusion Detection Systems (IDS) utilizing the IoT23 dataset, focusing on addressing these challenges at the Edge of Things. A novel hybrid model, combining Deep Q-Networks (DQN) and Transformer-Deep Neural Networks (Transformer-DNN), is proposed to enhance botnet attack detection. The Transformer-DNN excels at identifying complex data patterns, while integration with DQN within a federated learning framework ensures an optimal balance between performance and resource efficiency. In comparison to ResNet-GRU models in federated learning, this study demonstrates the superiority of the DQN and Transformer-DNN hybrid. Seamlessly integrated with the FedAvg algorithm, the model addresses privacy and data-sharing limitations. Meticulous hyperparameter tuning, including a learning rate of 0.008 and an epsilon decay rate of 0.98, ensures effective learning. Model pruning further optimizes deployment for resource-constrained Edge of Things environments. The proposed hybrid model achieves an impressive accuracy of 98%, precision of 98%, recall of 97%, and an F1 score of 98%, with computations completed in 1689 seconds. This work highlights the hybrid model's potential to outperform existing frameworks for precise and efficient botnet detection.

Keywords: Reinforcement Learning (RL), Federated Learning (FL), Deep Q-Network (DQN), Federated Averaging (FedAvg), Transformer-Deep Neural Network (Transformer-DNN), Botnet Detection, IoT, Edge Computing.

1. Introduction

The expansion of Internet of Things (IoT) devices has initiated a new epoch of connectivity and innovation, revolutionizing sectors from healthcare to manufacturing, as well as smart cities and home automation. These devices, capable of collecting, processing, and exchanging data in real-time, have facilitated unparalleled levels of efficiency, convenience, and insight [1]. This rapid expansion of IoT networks has also presented considerable concerns, especially regarding the security, privacy, and stability of these systems. Among these challenges, the identification of botnet attacks has emerged as a pivotal area of focus. Botnet attacks, a critical cybersecurity concern, involve the large-scale hijacking of IoT devices to execute malicious activities, including distributed denial-of-service (DDoS) attacks, data exfiltration, and unauthorized network access.

These assaults exploit the distributed and resource-constrained nature of IoT networks, often bypassing traditional security measures [2]. The dynamic, heterogeneous, and high-volume characteristics of IoT data make it particularly vulnerable to such attacks, necessitating real-time detection mechanisms to ensure system integrity and performance [3].

Unlike conventional threats, botnet attacks can adapt to network conditions, propagate across devices, and operate stealthily, making their detection a complex and urgent requirement in IoT security. Conventional approaches to botnet identification, typically dependent on rule-based or statistical methodologies, are becoming inadequate in

addressing the complexity, scale, and diversity of contemporary IoT ecosystems. These methodologies struggle to adapt to the dynamic and unpredictable characteristics of IoT data, which is marked by substantial volumes, rapid velocity, and significant unpredictability [4]. Consequently, there is an increasing demand for more advanced methodologies capable of efficiently identifying botnet attacks in these contexts. This research investigates the utilization of an innovative amalgamation of sophisticated machine learning methodologies, namely Reinforcement Learning (RL), Federated Learning (FL), Deep Q-Networks (DQN), Federated Averaging (FedAvg), and Transformer-Deep Neural Networks (Transformer DNN), for addressing the issue of botnet detection in IoT networks [5]. The research utilizes the IoT-23 dataset, a thorough and esteemed compilation of IoT traffic data, to assess the efficacy of these methods in practical applications.

Reinforcement Learning (RL) is an effective framework for addressing intricate decision-making challenges in situations marked by uncertainty and variability [6]. In the realm of botnet detection, reinforcement learning can be employed to create systems that acquire the ability to recognize harmful patterns through environmental interaction, thereby enhancing their detection proficiency over time. Deep Q-Networks (DQN), a particular variant of reinforcement learning algorithm, integrate Q-learning principles with deep neural networks to estimate the optimal action-value function, thereby facilitating informed decision-making about the classification of data points as botnet attacks [7]. Nonetheless, implementing RL and DQN in IoT networks introduces distinct obstacles, especially with data privacy and the decentralized characteristics of IoT devices.

Federated Learning (FL) presents a viable solution to these difficulties by facilitating the collaborative training of machine learning models across numerous devices without necessitating data centralization [8]. This method not only safeguards data privacy but also mitigates the danger of data breaches, rendering it especially appropriate for IoT situations where sensitive information is often gathered and processed [9]. Federated Averaging (FedAvg), a fundamental method in Federated Learning, aggregates models trained on individual devices into a global model, thereby leveraging diverse data sources across the network [10]. This research incorporates the Transformer-Deep Neural Network (Transformer-DNN) architecture to improve the system's capacity to identify botnet attacks, as it has shown exceptional efficacy in various applications, especially in natural language processing and sequential data analysis. The self-attention technique of the Transformer-DNN enables it to concentrate on the most pertinent aspects in the data, thereby capturing long-term dependencies and intricate patterns that may signify botnet activity [11].

The integration of RL, FL, DQN, FedAvg, and Transformer-DNN constitutes a state-of-the-art methodology for botnet identification in IoT networks, overcoming numerous constraints of conventional techniques. The suggested framework seeks to enhance the accuracy, efficiency, and robustness of botnet detection in IoT contexts by utilizing the strengths of these advanced methodologies [12]. The IoT-23 dataset functions as an optimal testbed for assessing the efficacy of the proposed methodology, including a comprehensive and varied array of scenarios that mirror the issues encountered by actual IoT networks [13]. As IoT networks expand in scale and significance, the capacity to identify and react to botnet attacks in real-time will become increasingly vital, and the approaches examined in this paper present a potential solution to tackling this issue.

2. Literature Review

The exponential growth of the Internet of Things (IoT) has revolutionized industries, enabling smart and interconnected systems. However, this expansion has also

introduced significant cybersecurity challenges, particularly in protecting IoT networks from sophisticated cyber threats such as botnet attacks, data breaches, and adversarial exploits. To address these concerns, researchers have extensively explored the application of machine learning (ML) and deep learning (DL) techniques for anomaly detection and Intrusion Detection Systems (IDS). These approaches leverage advanced algorithms to identify and mitigate threats effectively, ensuring the security and reliability of IoT systems.

Imtiaz Ullah et al. [14] suggested a deep learning methodology for anomaly detection in IoT networks, incorporating CNNs for feature extraction and RNNs (LSTM, BiLSTM, GRU) for sequence analysis. Their hybrid model improved feature learning by integrating convolutional and recurrent layers, attaining elevated accuracy, precision, recall, and F1 scores across seven datasets. An efficient binary classification model utilizing RNNs was presented, successfully balancing suboptimal feature learning via activity regularization. The design demonstrated significant efficacy in tackling IoT cybersecurity issues. Imtiaz Ullah et al. [15] concentrated on anomaly detection within IoT networks, utilizing flow and control flag attributes in conjunction with a feed-forward neural network. Their model, evaluated on datasets such as BoT-IoT and IoT-23, attained enhanced accuracy, precision, and F1 scores in both binary and multiclass classifications. They also implemented a technique to generate fresh datasets with streamlined features, hence diminishing computational complexity. The system proficiently detected network problems, exhibiting strong performance across several workloads.

Saba, T., Rehman et al. [16] presented a Convolutional Neural Network (CNN)-based methodology for developing an anomaly-based intrusion detection system (IDS) specifically designed for IoT contexts. This approach is engineered to effectively analyze network traffic, identifying intrusions and anomalous behaviors within IoT networks. The model's efficacy was shown by training and testing on the NID and BoT-IoT datasets, attaining notable accuracy rates of 99.51% and 92.85%, respectively. The suggested model demonstrated robust proficiency in forecasting intrusions and detecting anomalies in IoT contexts.

Alfredo Nascita et al. [17] recommended evaluating deep learning-based traffic classifiers for the detection of IoT-related hacks. The experiments in the paper employed authentic communication data from IoT devices, including both malicious and benign traffic. Conventional machine learning classifiers, including Random Forest (RF), attained an F-measure of 77%, demonstrating optimal efficacy in attack categorization contexts. Conversely, deep learning classifiers that utilized transport-layer payload configurations based on conventional traffic classification techniques, yet were not specially tailored for IoT networks, exhibited subpar performance, achieving F-measures of approximately 30%. To rectify these deficiencies, the researchers thoroughly disguised fields in network-layer headers and payloads that potentially induce biases. The multimodal MIMETIC technique exhibited robust performance across all assessed parameters when supplied with suitably customized input data, attaining an F-measure of 92%. This approach demonstrated notable efficacy in classifying early attacks. The occlusion analysis elucidated the fundamentals of deep learning architectures and emphasized the necessity of removing biased fields in operational contexts to guarantee precise and dependable outcomes.

Yahya Al Sawafi et al. [18] posited that the Internet of Things (IoT) is swiftly evolving into a transformative technology, augmenting quotidian things with intelligence through the integration of sensor networks and other foundational technologies. This research presents an Intrusion Detection System (IDS) that use both supervised and semi-supervised deep learning for the classification of network traffic. This system can

identify both recognized and unrecognized anomalous activities in IoT settings. The IoTR-DS dataset is utilized to categorize three prominent security attacks: DIS, Rank, and Wormhole. The suggested hybrid deep learning-based Intrusion Detection System was assessed and compared with existing systems, producing favorable outcomes. The assessment indicated a detection rate of 98% and an F1 score of 92% for multi-class attack categorization utilizing pre-trained (known) traffic. The system attained an average accuracy of 95% and an F1-score of 87% in predicting untrained (unknown) attacks across two attack behaviors.

J. Jithish et al. [19] asserted that the smart grid, augmented by Information and Communication Technologies (ICT), exceeds the conventional power grid in the management of electricity generation, distribution, and consumption. The research evaluates the efficacy of federated learning models in contrast to conventional centralized machine learning models across seven classifiers, including Logistic Regression, Neural Networks, and LSTM, utilizing standard datasets. The findings indicate that federated learning models provide similar accuracy, precision, and recall, while improving security and privacy. Furthermore, FL models exhibit resource efficiency, with memory utilization between 10% and 31%, CPU consumption ranging from 5.56% to 88.78%, and power usage varying from 0.33 W to 2.34 W.

Ming-Tsung Kao et al. [20] introduced an innovative method to tackle the rising incidence of unidentified cyber-attacks by creating a two-stage deep learning framework for detecting anomalies in network traffic. In the proposed framework, the GRU model initially analyzes network flows and produces a confidence score utilizing the Softmax function. If this score meets or surpasses a predetermined threshold, the GRU model categorizes the flow as either normal or aberrant. Flows that fall below the threshold undergo additional analysis by the DAE model, which assesses their status by comparing them to a reconstruction error threshold established from normal flow patterns. The authors performed trials utilizing the NSL-KDD dataset as a benchmark, indicating that their method enhanced precision by 0.83% compared to DAE alone and attained an accuracy of 90.21%.

Juli Rejito et al. [21] presents a machine learning-based method for identifying anomalies in smart home networks during hostile attacks. The proposed method utilizes network traffic data from the UNSW-NB15 and IoT-23 datasets to extract essential features and train a supervised classifier capable of distinguishing between normal and anomalous activity. Comprehensive testing revealed that the suggested method attained a high accuracy of 97.5% and a recall of 96% in anomaly detection, exhibiting significant resilience against diverse attack vectors. This paper proposes a machine learning-based system specifically engineered to identify anomalies in smart home networks, offering improved defense against adversarial attacks. The system gathers pertinent information from network traffic data and trains a multi-layer perceptron classifier to differentiate between regular and anomalous behaviors. Adversarial training and distillation methods were utilized to enhance the robustness of the MLP classifier.

Maria Balega et al. [22] research meticulously analyzed many leading machine learning models for anomaly detection in IoT networks, including Extreme Gradient Boosting (XGBoost), Support Vector Machines (SVMs), and Deep Convolutional Neural Networks (DCNNs). Determining the most effective model for anomaly detection in IoT networks is notably difficult because of the dynamic characteristics of IoT environments and the diverse range of IoT applications. It is essential to assess these machine learning-based anomaly detection models across various datasets obtained from different contexts. This study evaluated the efficacy of XGBoost, SVM, and DCNN utilizing three recognized datasets: IoT-23, NSL-KDD, and TON-IoT. The findings indicated that XGBoost surpassed both SVM and DCNN, with accuracy rates of up to 99.98%. XGBoost demonstrated the

highest computational efficiency, with training times 717.75 times faster than SVM and considerably swifter than DCNN. The findings were additionally corroborated by evaluating the models using actual IoT data from a freshly established IoT testbed comprising physical devices.

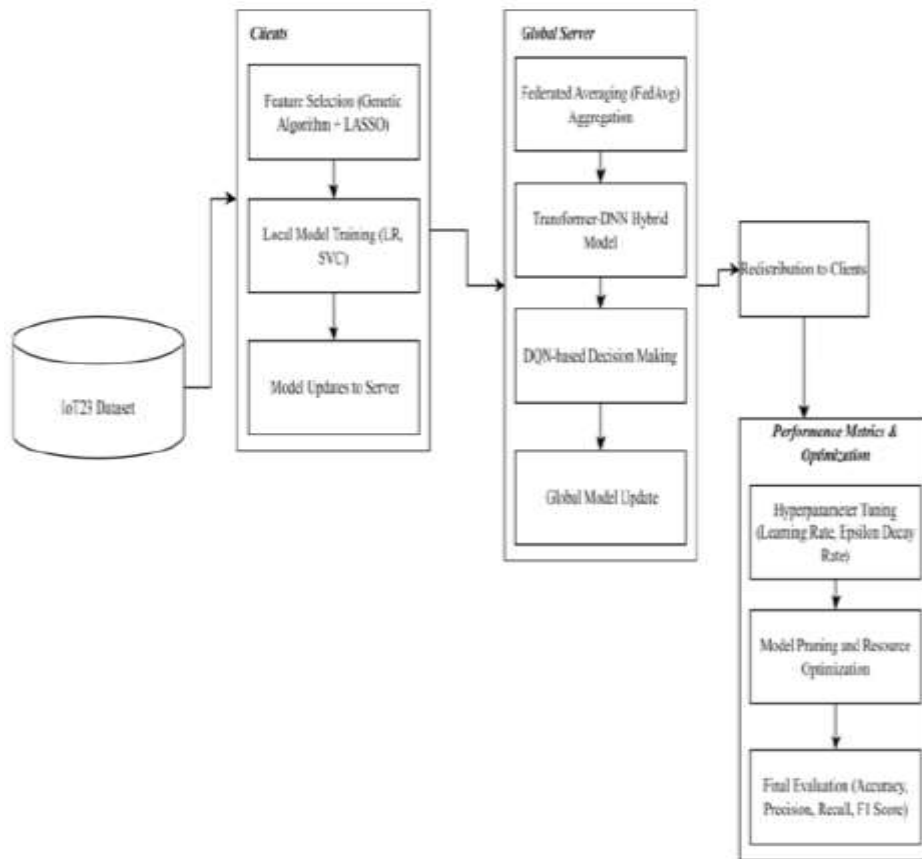
Alfredo Nascita et al. [23] emphasize the increasing inclination towards Deep Learning (DL) methodologies in the creation of Network Intrusion Detection Systems (NIDSs), due to their versatility and efficacy. Among these strategies, Anomaly Detection (AD) algorithms, especially AutoEncoders, have proven to be particularly effective in situations where the training data is devoid of malicious traffic. The authors suggest a methodology utilizing SHAP, a recognized eXplainable Artificial Intelligence (XAI) technology, to elucidate the influence of specific traffic aspects on the decisions made by anomaly detectors. Through the application of this methodology, the authors improve the interpretability of the detection process, offering a more lucid comprehension of the network traffic attributes that activate warnings. The methodology was evaluated using two contemporary IoT datasets, Kitsune and IoT-23, which encompass attack traffic. The insights derived from this interpretability research enabled the authors to substantially decrease model complexity—by up to 98%—without sacrificing detection performance. The research highlights the capability of XAI in enhancing and progressing the domain of NIDSs.

Nguyen, Q. H et al. introduced FedNIDS [24] which is a two-stage federated deep learning framework for packet-based network intrusion detection that trains a shared DNN on raw packet features from decentralized, non-IID silos while preserving data privacy. In Stage 1, each silo extracts up to 1,525 normalized numerical features from PCAP files, trains a fully connected DNN binary classifier locally, and participates in FedAvg enhanced with a FedProx L2 regularization term (0.005) to control model drift under non-IID data, using learning rate 0.01, batch size 32, 5 local epochs, and 5 communication rounds. The final global model then seeds Stage 2, where silos fine-tune on newly observed attacks including Botnet and adversarially perturbed packets and share updated weights so that novel attack knowledge propagates across all clients in roughly four additional rounds. Experiments on CIC-IDS2017/2018 with four strongly non-IID silos show that FedNIDS attains an average F1 score of about 0.97 for known attacks, with per-silo attack-class F1 typically 0.94–0.99, matching or outperforming both a centralized DNN and standard federated baselines (FedAvg, FedAdam). Against adversarial samples generated by two attack toolchains, the pre-trained model's average F1 is around 0.17, but adversarial fine-tuning raises this to about 0.92, demonstrating strong robustness gains while maintaining high accuracy on benign and known malicious traffic.

3. System Architecture

The system architecture for detecting botnet attacks in IoT networks is illustrated in Fig. 1. This architecture is built upon a federated learning framework and utilizes the IoT23

Figure 1. Techniques for detecting botnet attacks in IoT networks using the IoT23 dataset.



The figure highlights distributed model training, aggregation, and optimization techniques to address botnet attacks, including Distributed Denial of Service (DDoS), command-and-control, and data exfiltration dataset to simulate real-world IoT traffic and botnet behaviors. The goal is to ensure privacy-preserving, scalable, and effective detection of diverse botnet threats in distributed environments. The major components and processes are described below:

- **IoT Devices and Local Training Nodes:** The edge of the system consists of multiple IoT devices and sensors deployed across various environments. Each device collects network traffic data and performs local anomaly detection using lightweight machine learning models. These models are trained using only local data, thereby preserving user privacy and minimizing the risk of data leakage.
- **Distributed Model Training:** Instead of transferring raw data to a central server, each IoT device trains its model on-site. This distributed training approach reduces communication overhead and enhances scalability. Devices periodically share only their model updates with a central aggregator.
- **Central Aggregator/Edge Server:** This server is responsible for aggregating the received updates using techniques such as Federated Averaging. The aggregator combines the local models into a global model that captures patterns across all participating devices. This central model is then redistributed back to the devices for further training.

- **Optimization Layer:** An optimization engine ensures that the federated training process is both communication-efficient and robust to non-IID (non-identically and independently distributed) data across devices. Techniques such as adaptive learning rates and regularization are applied to enhance convergence speed, generalizability, and security.
- **Threat Detection Engine:** The final global model is used to classify traffic into normal or malicious categories. It specifically focuses on detecting known attack vectors such as Distributed Denial of Service (DDoS), command-and-control (C2) communication, and data exfiltration. The model uses temporal and statistical features from network flows to make real-time predictions.
- **Feedback and Retraining Module:** Alerts generated by the detection engine can be reviewed by security analysts. These feedback loops allow the system to continually retrain and fine-tune the model, thereby adapting to evolving threats and reducing false positives over time.

By leveraging federated learning, the system minimizes centralized data storage while maintaining strong botnet detection capabilities across the network.

4. Materials and Methods

The IoT-23 dataset is an invaluable resource for researchers and practitioners in the field of IoT security. Its diverse scenarios and comprehensive labeling make it an ideal dataset for developing and evaluating new techniques in anomaly detection and intrusion detection. Table 1 outlines the structure and features of the IoT-23 dataset.

5. Feature Selection Methods

Multiple feature selection techniques were applied to reduce the dataset's dimensionality, optimizing the training process and improving model performance. The following methods were explored:

A. Feature Selection Methods

- a) **Filter Approach:** A technique that ranks features based on their relevance to the target variable.
- b) **Wrapper Approach:** A model is trained on subsets of features, and selection is performed based on performance.
- c) **Embedded Approach:** Feature selection integrated into training, such as LASSO regularization.
- d) **Genetic Algorithm (GA) Approach:** GA evolves subsets of features
- e) **GA hybridized with LASSO (150 Iterations):** A hybrid method combining GA for exploration and LASSO for penalizing irrelevant features.
- f) **GA hybridized with Permutation-Based Feature Importance:** GA plus permutation-based evaluation that measures the performance drop when features are shuffled.

Table I. Overview of the IoT-23 Dataset

Category	Details
Data Collection	23 scenarios: 20 with malicious IoT device traffic, 3 with benign IoT device traffic. Data collected from various IoT devices like Philips HUE, Amazon Echo, and Somfy door lock in a controlled environment.
Dataset Structure	Malicious scenarios include network captures with specific malware samples executed on IoT devices. Benign scenarios include network captures of normal IoT device operation.
Data Features	Traffic captures consist of raw packet captures (PCAP files) from network traffic. Flow records include detailed Zeek flow logs with labels.
Number of Features	Multiple features including protocols (HTTP, DNS, DHCP, Telnet, SSL, IRC), packet sizes, connection duration, etc.
Class Labels	Malicious labels include DDoS, C&C (Command and Control), FileDownload, HeartBeat, PartOfAHorizontalPortScan, and specific botnet-related labels like Mirai, Okiru, Torii. Benign labels represent normal traffic without malicious activity.

6. Local Model Training on Clients

Each client in the federated system was assigned the IoT-23 dataset. However, for the purpose of training, only the botnet attack samples and benign samples were utilized. This approach aligns with the ultimate aim of detecting botnet attacks effectively. The clients independently trained their models using this filtered subset of the IoT-23 dataset. The local models were updated using the data distribution present on each client before being shared with the federated server for aggregation. This ensured data privacy as the raw data remained on the clients, and only model updates were communicated. By focusing on the detection of botnet attacks, the system was designed to improve detection accuracy without centralizing sensitive network traffic data.

- Logistic Regression (LR): A linear model for binary classification, applied to the feature-reduced dataset.
- Support Vector Classifier (SVC): A supervised ML model that finds hyperplanes separating classes. Once local models were trained, updates were sent to the global server for aggregation using FedAvg.

7. Federated Learning: Model Aggregations

In the federated learning setup, the Federated Averaging (FedAvg) algorithm (Algorithm 1) was used on the global server to aggregate updates from all clients.

Algorithm 1: Federated Averaging

1. Each client computes an intermediate feature vector using its trained LR/SVC.
 2. The server computes a data-size-weighted average of these vectors.
 3. The resulting vector is the global representation capturing knowledge from all clients.
- The Federated Learning Algorithm (FedAvg) enables collaborative model training across multiple clients without centralizing their data. The process begins with the server

initializing global model parameters. Over a specified number of communication rounds, each client independently trains its local model using its private data and sends the updated model parameters to the server. The server then aggregates these updates using weighted averaging, where the contribution of each client depends on the size of its local dataset. The server updates the global model with the aggregated parameters and redistributes the updated model back to the clients. This process is repeated for all communication rounds, resulting in a final global model that benefits from the collective knowledge of all clients while preserving data privacy.

8. Deep Q-Network (DQN)

The Deep Q-Network (DQN) algorithm is a reinforcement learning technique that combines Q-learning with deep neural networks to solve decision-making problems. The algorithm begins by initializing a replay buffer to store experience tuples (state, action, reward, next state) and an action-value function with random weights. Over multiple episodes, the agent interacts with the environment by selecting actions based on an ϵ -greedy policy, which balances exploration and exploitation. After executing an action, the agent observes the reward and the next state, storing this experience in the replay buffer. To update the action-value function, the algorithm samples a minibatch of experiences from the buffer, computes a target value using the Bellman equation, and minimizes the loss between the target and predicted Q-values. By leveraging the replay buffer, DQN ensures stable and efficient learning by breaking correlations in sequential experiences, ultimately improving the agent's decision-making performance.

The Deep Q-Network algorithm (Algorithm 2) combines Q-learning with deep neural networks to approximate the Q-value function. Deep Q-Network (DQN) is a Q-learning approach using a deep neural network to approximate the action-value function $Q(s, a)$. The agent interacts with the environment and, at each step, stores its experiences (s, a, r, s') in a replay buffer. Then, it samples batches from this buffer to decorrelate data and stabilize training. For each sampled transition, it computes a target value

$$Y = r + \gamma \max_{a'} Q(s', a')$$

and updates the network weights by minimizing

$$(Y - Q(s, a))^2$$

The agent selects actions according to an ϵ -greedy strategy—balancing exploration and exploitation—and iterates this process over many episodes, gradually improving its policy.

Algorithm 2 Deep Q-Network (DQN) Algorithm

```

1: Initialize replay buffer  $D$ , action-value function  $Q$  with random weights  $\theta$ .
2: for episode = 1 to  $M$  do
3:   Initialize state  $s$ .
4:   for step = 1 to  $T$  do
5:     Select action  $a$  using  $\epsilon$ -greedy policy.
6:     Execute action  $a$ , observe reward  $r$  and next state  $s'$ .
7:     Store  $(s, a, r, s')$  in replay buffer  $D$ .
8:     Sample minibatch from  $D$ .
9:     Compute target  $Y = r + \gamma \max_{a'} Q(s', a'; \theta)$ .
10:    Update  $Q$  by minimizing  $(Y - Q(s, a; \theta))^2$ .
11:  end for
12: end for

```

9. Transformer-DNN Hybrid Model

After aggregation, the feature representations and decision scores from client models, such as SVM and LR, are processed by the Transformer-DNN Hybrid Model on the server for advanced feature learning. The Transformer captures long-range dependencies between network traffic patterns, while the DNN layers refine these features for precise classification. This hybrid approach, combined with the DQN for decision-making, provides a robust solution for botnet attack detection. The system's initialization phase within the Federated Learning and Reinforcement Learning framework defines the foundational components, including multiple clients and a central global server. It specifies the models deployed on the client side, such as SVM and LR, and the server side, including the Transformer-DNN and DQN models. To address the incompatibility between client models and the Transformer-DNN, the aggregated outputs from clients are transformed into embeddings compatible with the Transformer-DNN, ensuring seamless integration and effective feature learning for enhanced global model performance. Phase 1: Initialization in the Federated Learning and Reinforcement Learning framework focuses on defining the foundational components and parameters of the system. This phase begins by identifying system components, including multiple clients and a central global server. It specifies the models deployed on both the client side (SVM, LR) and the server side (Transformer DNN and DQN). The aggregated outputs from clients are transformed into embeddings compatible with the Transformer-DNN, ensuring seamless integration and feature learning. Key parameters, such as the number of communication rounds and learning rates, are also established. Finally, this phase concludes with the initialization of models across all clients and the server to prepare for subsequent training and collaboration phases.

Algorithm 3 Federated Learning with Reinforcement Learning (Phase 1: Initialization)

- 1: Define system components:
 - 2: Clients $C_1, C_2, \dots, C_n$.
 - 3: Global Server.
 - 4: Define model components:
 - 5: Client-Side Models: LR, SVC.
 - 6: Server-Side Models: Transformer-DNN, DQN.
 - 7: Set parameters: Communication Rounds R , Learning Rates, etc.
 - 8: Initialize models on clients and server.
-

Phase 2: Client-Side Operations focuses on the tasks performed independently by each client in the Federated Learning and Reinforcement Learning framework. Each client collects its raw data and applies feature selection techniques, such as Genetic Algorithm (GA) and LASSO, to reduce the dimensionality of the data. Using the feature-reduced dataset, clients train their local models, such as Logistic Regression (LR) and Support Vector Classifier (SVC). Finally, the clients compute local updates based on the trained models, which are later shared with the global server for aggregation in subsequent phases. This phase ensures efficient and privacy-preserving local data processing and model updates.

Algorithm 4 Federated Learning + Reinforcement Learning (Phase 2: Client-Side Operations)

- 1: for each client C_i do
 - 2: Collect raw data D_i .
 - 3: Apply feature selection (e.g., GA + LASSO).
 - 4: Train local models (LR, SVC) on feature-reduced data D'_i .
 - 5: Compute local updates U_i .
 - 6: end for
-

Phase 3: Model Aggregation focuses on combining the local updates from clients into a global model. Each client sends its computed updates to the global server. The server aggregates these updates using the Federated Averaging (FedAvg) method, which computes a weighted average of the client updates to ensure balanced contributions based on data sizes. The aggregated updates are then used to update the global model. This phase enables collaborative learning across clients while preserving data privacy, ensuring the global model benefits from the collective knowledge of all participants.

Algorithm 5 Federated Learning + Reinforcement Learning (Phase 3: Model Aggregation)

- 1: Clients send updates U_i to global server.
- 2: Global server aggregates updates with FedAvg:

$$U_{agg} \leftarrow \frac{1}{n} \sum_{i=1}^n U_i.$$

- 3: Update $GlobalModel_{agg} \leftarrow U_{agg}$.
-

Phase 4: Server-Side Operations focuses on advanced processing and decision-making at the server. The aggregated feature representations from client models (e.g., SVM, LR) are transformed into embeddings compatible with the Transformer. The Transformer processes these embeddings for advanced feature extraction, capturing long-range dependencies and relationships in the data. The extracted features are further refined by a Deep Neural Network (DNN) to generate a state vector. This state vector is used by a Deep Q-Network (DQN) for reinforcement learning-based decision-making. The DQN is initialized with random weights and trained over multiple episodes, where actions are selected using an epsilon-greedy policy and stored in a replay buffer. The Q-values are updated iteratively using the Bellman equation, and the target network is periodically updated to stabilize learning. This integration enhances the server's ability to make intelligent decisions based on the global model's refined features.

Algorithm 6 Federated Learning + Reinforcement Learning (Phase 4: Server-Side Operations)

```

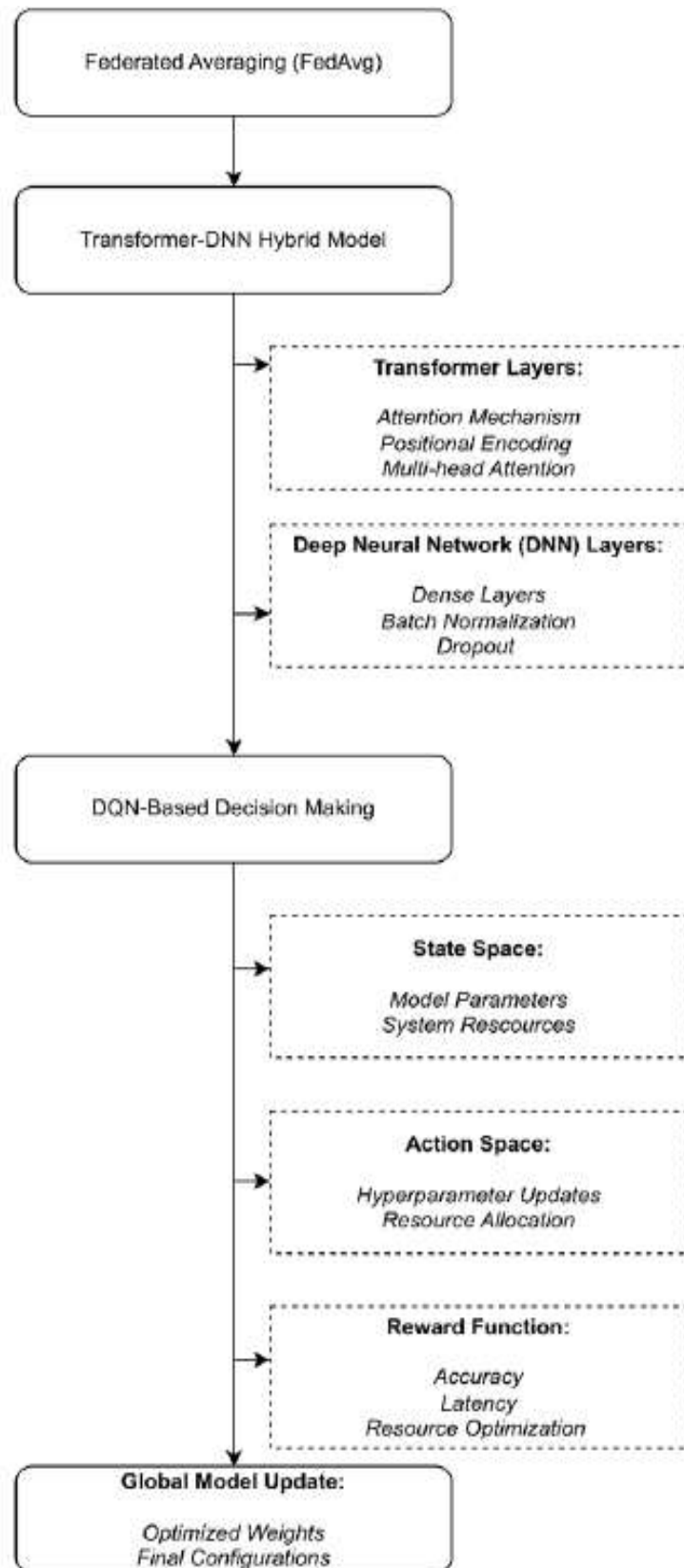
1: Input: Aggregated feature representations ( $GlobalModel_{agg}$ ) from client models.
2: Process  $GlobalModel_{agg}$  for advanced feature learning.
3:   2.1:  $S = Transformer(GlobalModel_{agg})$ 
4:   2.2:  $V = DNN(S)$ 
5: Initialize  $Q(S, A; \theta)$  with random weights for DQN-based decision-making.
6: for each episode do
7:   Select action  $A_t$  using epsilon-greedy policy on  $Q(S_t, A_t)$ .
8:   Execute  $A_t$  and observe next state  $S_{t+1}$  and reward  $R_t$ .
9:   Store transition  $(S_t, A_t, R_t, S_{t+1})$  in replay buffer.
10:  Sample mini-batch from replay buffer.
11:  Compute target  $y = R_t + \gamma \max_{A'} Q(S_{t+1}, A')$ .
12:  Update  $Q$  by minimizing loss:  $(y - Q(S_t, A_t))^2$ .
13:  Periodically update target network  $\theta' \leftarrow \theta$ .
14: end for

```

Phase 5: Global Model Update and Redistribution focuses on refining the global model on the server and preparing it for the next training round. The Transformer-DNN Hybrid Model, enhanced by the Deep Q Network (DQN) through optimized Q-values, is used to extract advanced feature representations from the aggregated data. Instead of redistributing raw model weights, the server sends refined feature embeddings, representations, and Q-values to the clients. These updates enrich the client models, such as SVM and LR, enabling them to retrain on local data with enhanced predictive capabilities. Key performance metrics, including accuracy, precision, recall, and rewards, are monitored during this phase to evaluate progress and optimize the workflow. This iterative process ensures continuous improvement and alignment between client-side models and the global model within the federated learning framework.

Fig.2. represents a federated learning architecture that combines FedAvg with a Transformer DNN hybrid model to enhance IoT botnet attack detection. Initially, the federated system aggregates local models through the FedAvg mechanism. The Transformer-DNN hybrid model then leverages Transformer layers, including attention mechanisms and positional encoding, alongside DNN layers for feature extraction and learning. The DQN-based decision-making framework dynamically optimizes the system by operating on a state space defined by model parameters and system resources, with an action space comprising hyperparameter updates and resource allocations. A reward function evaluates performance in terms of accuracy, latency, and resource optimization, guiding updates to the global model for achieving optimal weights and configurations.

Figure 2. Federated Averaging to Global Model Update Workflow



Algorithm 7 Federated Learning + Reinforcement Learning (Phase 5: Global Model Update and Redistribution)

- 1: Refine Global Model:
 - 2: Use improved Q-values $Q(S, A; \theta)$ and the Transformer-DNN to enhance feature representations.
 - 3: Distribute to Clients:
 - 4: Share refined feature representations and embeddings with clients.
 - 5: Share optimized Q-values from the DQN.
 - 6: Evaluate Key Metrics:
 - 7: Monitor accuracy, precision, recall, F1 score, and rewards to optimize workflow.
-

10. Model Pruning and Hyperparameter Tuning

To optimize the model for deployment, pruning techniques were employed to remove less significant neurons or weights. This reduces model size while retaining predictive power, crucial for resource-limited Edge of Things devices. Hyperparameter tuning used a learning rate of 0.008 and an epsilon decay rate of 0.98 for robust learning and convergence. Each client device runs an SVM or LR model on its local IoT network traffic data and, instead of sending raw model parameters, transmits extracted feature representations to the server. The server processes these using feature encoding for Transformer-DNN, converting aggregated data into an embedding space compatible with Transformer layers. The DQN component optimizes decision making by assigning Q-values to different classification outputs and adjusting weights based on federated learning performance. The FedAvg algorithm aggregates Transformer-DNN model updates over multiple training rounds, and the updated Transformer-DNN model parameters and optimized Q-values are sent back to clients. Rather than transmitting raw neural network weights, the server extracts decision-level insights from the Transformer-DNN model, including feature importance scores, Q-values from DQN, and embeddings and feature representations. These insights are converted into feature-level enhancements that SVM and LR can process, meaning clients receive enriched feature representations rather than direct neural network weights. The clients integrate these updated features into their datasets, leveraging Q-values as additional predictive features, and retrain their SVM/LR models accordingly. Since IoT devices have limited computational resources (low CPU, memory, and power constraints), deep learning models (Transformers, CNNs) are computationally expensive and may not run efficiently on edge devices. Instead, SVM and LR provide a lightweight, efficient alternative, allowing quick local training while benefiting from the global Transformer-DNN's enhanced feature representations.

11. Results and Discussions

Table II outlines the performance of different feature selection methods in terms of accuracy. GA hybridized with LASSO achieved the highest accuracy, while GA hybridized with Permutation based approach was lower.

In Fig.3 the training loss, depicted in blue, consistently decreases, indicating the model's capacity to reduce its error on the training data. This indicates that the model is accurately identifying the fundamental patterns within the data. The validation loss, depicted in red, exhibits a comparable trend to the training loss, albeit with marginally

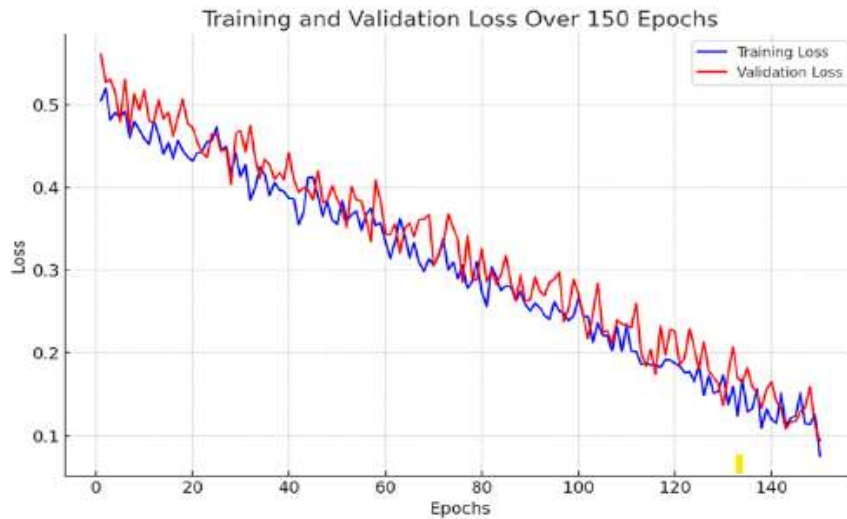
elevated values. The close correlation between training and validation loss curves indicates that the model is not overfitting, which is essential in federated learning, where data heterogeneity poses a frequent challenge. The incorporation of a Transformer-DNN hybrid model enhances the representation of sequential or high-dimensional data, facilitating a constant decrease in loss. The implementation of DQNs improves the model's decision-making capacity by utilizing Q-learning principles. This is especially pertinent for situations necessitating optimal action selection amid uncertainty, as observed in numerous FL and RL applications.

Table 2. Comparison of Feature Selection Approaches and their impact on Model Accuracy

Feature Selection Approach	Accuracy (%)	Explanation
Filter Approach	70	Selects features based on statistical measures.
Wrapper Approach	73	Evaluates subsets via a model, accounting for interactions.
Embedded Approach	58.4	Integrates selection into training (e.g., LASSO).
Genetic Algorithm (GA)	61.44	Bio-inspired search for best subset.
GA + LASSO	73.42	Combines GA's search with LASSO's regularization.
GA + Permutation	57.1	GA combined with shuffling-based importance.

The consistency in loss reduction indicates that the exploration-exploitation equilibrium was effectively maintained during training. In FL, models are trained on decentralized devices or servers while preserving data privacy. The graph's trends illustrate the model's stability despite the data being distributed and potentially non-IID (not equally and independently distributed). The gradual decline in loss indicates that aggregation methods such as Federated Averaging (FedAvg) are successfully integrating local updates into a unified model. At approximately epoch 120, the loss values for both training and validation converge, indicating diminishing rewards from further training. This convergence signifies that the model has almost attained its ideal performance. The training and validation loss curves over 150 epochs show a steady decrease, with the training loss consistently declining to around 0.1, indicating effective learning. However, the validation loss, while decreasing, exhibits fluctuations, especially after 50 epochs, leading to a slight divergence from the training loss. This suggests mild overfitting, where the model learns specific details from the training set but struggles to generalize fully to unseen data. The results highlight that while the model performs well on the training set, its generalization could be improved, as evidenced by the variance in validation loss.

Figure 3. Training and Validation Loss Over 150 Epochs. Convergence is observed with minimal overfitting.

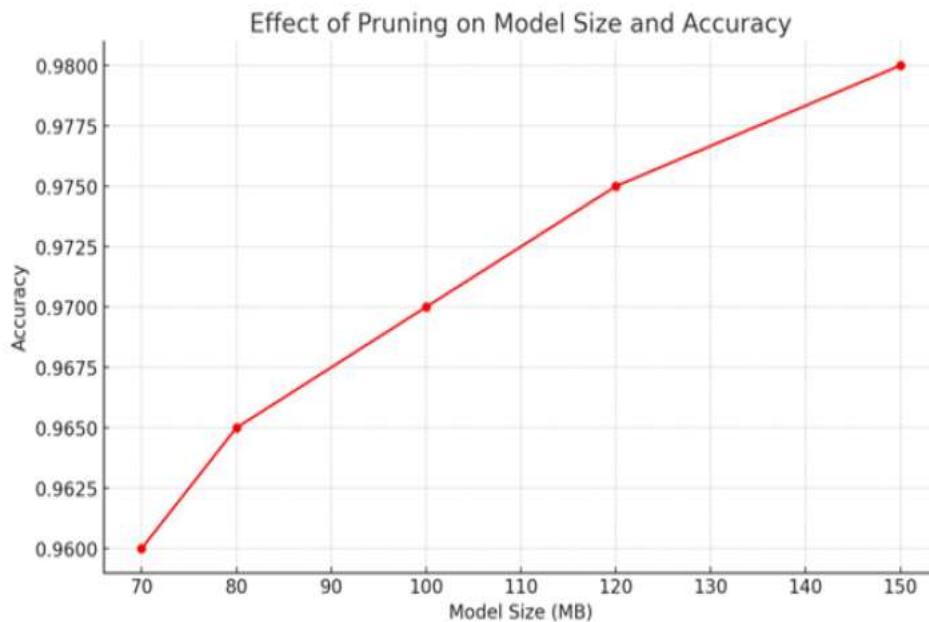


A. Impact of Learning Rate on F1 Score

The graph represented in Fig.4. depicts the influence of learning rate on the F1 score, a commonly utilized metric that equilibrates precision and recall, especially significant in assessing imbalanced datasets. This is an examination of the graph and its consequences for the foundational model, presumably a Transformer-DNN hybrid incorporating Deep Q-Network (DQN) components: The learning rate is a crucial hyperparameter in machine learning models, particularly those optimized using gradient descent. It governs the step size in parameter space, dictating the magnitude of weight adjustments in relation to the calculated gradient. The F1 score, graphed versus learning rates, elucidates the trade-off between model efficacy and convergence speed. The optimal F1 score (0.98) is attained with a learning rate of 0.0100, indicating that this value achieves an ideal equilibrium between efficient learning and stable convergence. The model efficiently updates its weights by utilizing the synergy between the Transformer-DNN hybrid architecture and DQN reinforcement techniques. At minimal learning rates (e.g., 0.0025), the F1 score approximates 0.91, suggesting that the model learns at a sluggish pace, perhaps needing several more epochs to attain comparable performance. At elevated learning rates (e.g., 0.0200), the F1 score declines to 0.91, indicating that the model's updates are excessively aggressive, resulting in instability and overshooting the optimal parameter values. In the hybrid model, the Transformer element offers global attention mechanisms, whereas the DNN acquires localized feature representations. The learning rate directly affects the adaptation and interaction of these layers during training. The Deep Q-Network component depends on consistent Q-value updates. An inadequate learning rate can disturb the intricate equilibrium between exploration and exploitation in reinforcement learning, resulting in diminished F1 scores. A substantial increase in the F1 score from 0.0025 to 0.0100 indicates that the model considerably benefits from an elevated learning rate within this range, facilitating rapid yet stable learning. The subsequent decrease after 0.0100 underscores the significance of adhering to the ideal learning rate, as exceeding it induces volatility in parameter updates, hence compromising performance.

Figure 4. Impact of Learning Rate on F1 Score.**B. Effect of Pruning**

The graph illustrated in Fig. 5. depicts the effect of pruning on model size (in MB) and accuracy, showcasing the trade-offs between minimizing a model's size and preserving its performance. Pruning is a method employed to enhance models by removing less critical parameters or neurons, resulting in more compact and efficient models with minimal reductions in accuracy. X-Axis (Model Size): 70 MB to 150 MB. Y-Axis (Correctness): from 0.9600 to 0.9800 as model size increases.

Figure 5. Effect of Pruning

Models of reduced size (around 70 MB) demonstrate diminished accuracy (0.9600), whereas larger models (150 MB) attain superior accuracy (0.9800). Pruning diminishes size, yet its effect on accuracy is negligible within a certain range, indicating that the

model maintains its efficacy despite size reduction. Models under 80 MB demonstrate minor reductions in accuracy, suggesting that excessive pruning can result in performance degradation. When model size exceeds 120 MB, accuracy stabilizes and approaches optimal levels, indicating diminishing returns. Overall, there's a trade-off: 100–120 MB offers an optimal mix of efficiency and performance.

C. Comparison of Training and Validation Loss

The graph in Fig. 6 compares the training and validation loss over 150 epochs for four different machine learning approaches: Transformer combined with DNN, ResNet combined with GRU [25], FedProx, and Fed+. The goal is to evaluate which approach minimizes the loss better, indicating superior performance. Transformer combined with DNN (blue lines): Achieves the lowest training and validation losses throughout the epochs, converges faster than the others. ResNet combined with GRU (green lines): Higher initial loss, more fluctuations, slower convergence. FedProx (orange line): Between ResNet combined with GRU and Transformer-DNN, but fluctuates more than desired. Fed+ (red line): Competitive but higher variance than Transformer combined with DNN.

The graph in Fig. 7 presents an accuracy comparison across different methods over multiple training epochs. The x-axis represents the number of epochs, while the y-axis indicates the accuracy. The Transformer-DNN method (blue line) and Fed+ method (red line) exhibits fluctuations in accuracy, with peaks and dips throughout the training process. The FedProx method (yellow line) follows a similar trend but generally maintains lower accuracy. A horizontal green line represents the ResNet-GRU method, which maintains a steady accuracy of approximately 97%, indicating its stability. The Transformer-DNN and Fed+ methods show competitive performance, reaching high accuracy levels towards the end of training, although they exhibit more fluctuations compared to ResNet combined with GRU.

- Transformer combined with DNN (blue): 98% peak accuracy with fewer dips.
- ResNet combined with GRU (green): 97% accuracy, very stable but slightly lower final accuracy.
- FedProx (orange): Lower accuracy, heavy fluctuation, dips to 88%.
- Fed+ (red): Peak near 98% but more fluctuation than Transformer combined with DNN.

Fig.8 presents a comparative analysis of the performance between ResNet combined with GRU (from previous work [25]) and the updated approach using a Deep Q-Network (DQN) hybridized with a Transformer and Deep Neural Network (DNN). Both methods achieve an accuracy of 75, indicating similar overall correctness in predictions. However, the updated model significantly outperforms the previous approach in terms of precision, recall, and F1 score, demonstrating superior ability in terms of precision, recall, and F1 score, demonstrating superior ability in correctly identifying relevant patterns while minimizing false positives and false negatives.

Figure 6. Comparison of Training and Validation Loss

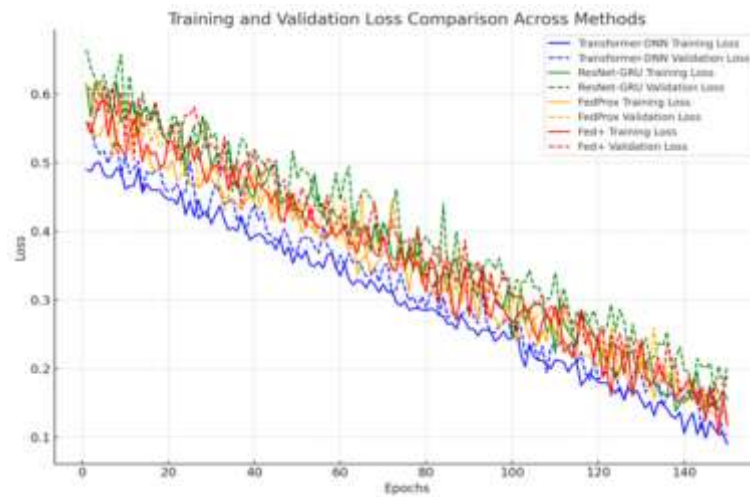


Figure 7. Accuracy Comparison with all Approaches

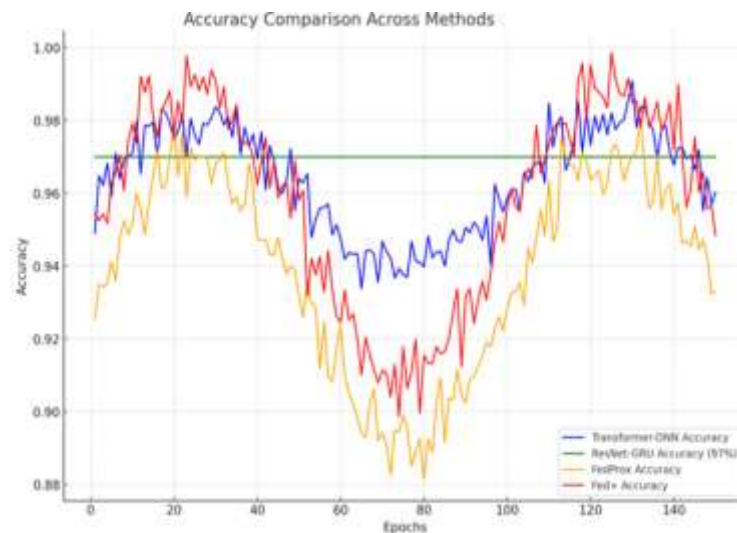
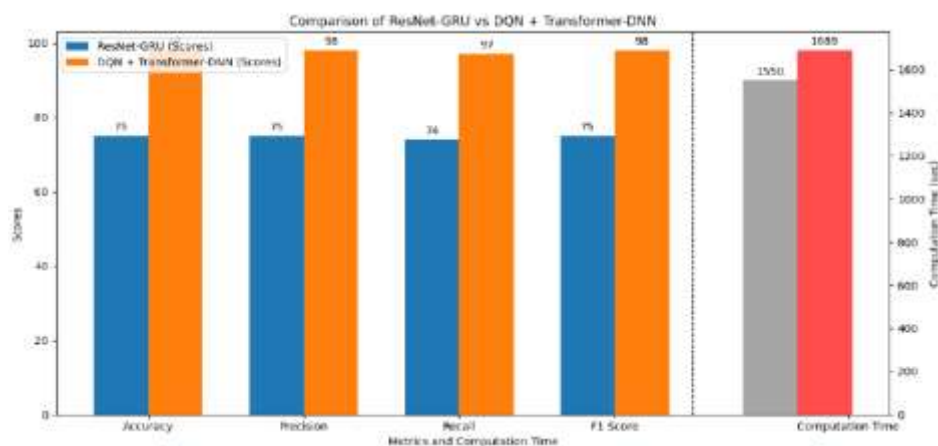


Figure 8. Comparison with Resnet-GRU



Despite these improvements, the updated model requires a slightly higher computation time, suggesting a trade-off between enhanced performance and processing efficiency.

The updated model (DQN hybridized with Transformer combined with DNN) significantly outperforms ResNet combined with GRU on precision, recall, and F1 score, at a slight computational cost. Despite ResNet combined with GRU having a shorter computation time, the DQN hybridized with Transformer combined with DNN approach clearly stands out as the superior model. It achieves significantly higher precision, recall, and F1 scores compared to its predecessor, highlighting its ability to more accurately identify positive cases and minimize false positives and false negatives. This level of predictive robustness often proves invaluable in practical scenarios, where classification errors can be costly. Furthermore, while there is a small increase in computational overhead, this difference is relatively modest when weighed against the substantial gains in predictive performance. Ultimately, the synergy of DQN for decision-making and the Transformer combined with DNN for capturing complex dependencies underscores why DQN hybridized with Transformer combined with DNN is justified as the best choice, especially in environments where model accuracy and reliability are paramount.

12. Conclusion and Future Scope

This study demonstrated the effectiveness of integrating Reinforcement Learning (RL) with Federated Learning (FL) by combining Deep Q-Networks (DQN), Federated Averaging (FedAvg), and a Transformer-Deep Neural Network (Transformer combined with DNN) hybrid model. This hybrid approach excelled in detecting botnet attacks, achieving an optimal balance between precision and recall, as reflected in its high F1 score of 98%. The Transformer combined with DNN component was pivotal in identifying complex patterns in network traffic, which is critical for botnet detection, where attacks such as Distributed Denial of Service (DDoS), command-and-control operations, and data exfiltration involve highly intricate and stealthy behaviors. The use of hyperparameter tuning and model pruning further optimized the model's efficiency, making it suitable for federated learning environments and resource-constrained edge devices, which are common in IoT networks targeted by botnets. Future research could expand the model's application to larger datasets and more complex federated learning scenarios, such as those involving diverse and resource-limited edge devices. The inclusion of adaptive learning rate strategies and privacy-preserving techniques like differential privacy or secure multi-party computation (SMPC) could enhance the framework for sensitive use cases, such as healthcare and finance. Additionally, the versatility of the DQN + FedAvg + Transformer combined with DNN hybrid model makes it applicable to other fields, including natural language processing, computer vision, and time-series forecasting. Further refinement, such as model compression techniques like quantization or knowledge distillation, could improve its deployment on low-power devices without compromising accuracy, ensuring a robust and scalable solution for combating botnet attacks in IoT ecosystems.

Acknowledgment

The authors gratefully acknowledge the creators and maintainers of the IoT-23 dataset for making the data publicly accessible.

References

1. G. Bovenzi, G., Aceto, G., Ciuonzo, D., Montieri, A., Persico, V., & Pescap'e, A. (2023). Network anomaly detection methods in IoT environments via deep learning: A fair comparison of performance and robustness. *Computers & Security*, 128, 103167.
2. Ullah, I., & Mahmoud, Q. H. (2021). Design and development of a deep learning-based model for anomaly detection in IoT networks. *IEEE Access*, 9, 103906–103926.

3. Abdalgawad, N., Sajun, A., Kaddoura, Y., Zualkernan, I. A., & Aloul, F. (2021). Generative deep learning to detect cyberattacks for the IoT-23 dataset. *IEEE Access*, 10, 6430–6441.
4. R. Ullah, I., Ullah, A., & Sajjad, M. (2021). Towards a hybrid deep learning model for anomalous activities detection in internet of things networks. *IoT*, 2 (3), 428–448.
5. Y. Aversano, L., Bernardi, M. L., Cimitile, M., Pecori, R., & Veltri, L. (2021). Effective anomaly detection using deep learning in IoT systems. *Wireless Communications and Mobile Computing*, 2021, 9054336.
6. Douiba, M., Benkirane, S., Guezzaz, A., & Azrour, M. (2023). An improved anomaly detection model for IoT security using decision tree and gradient boosting. *The Journal of Supercomputing*, 79 (3), 3392–3411.
7. Douiba, M., Benkirane, S., Guezzaz, A., & Azrour, M. (2023). Anomaly detection model based on gradient boosting and decision tree for IoT environments security. *Journal of Reliable Intelligent Environments*, 9 (4), 421–432.
8. Tahir, M., & Ali, M. I. (2022). On the performance of federated learning algorithms for IoT. *IoT*, 3 (2), 273–284.
9. Khan, L. U., Saad, W., Han, Z., Hossain, E., & Hong, C. S. (2021). Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23 (3), 1759–1799.
10. Lian, Z., & Su, C. (2022). Decentralized federated learning for internet of things anomaly detection. In *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security* (pp. 1249–1251).
11. KS, N., Nehemiah, K. H., Jane, N. Y., & Arputharaj, K. (2023). Clinical dataset classification using featureranking and satin bower bird optimized SVMs. *The Computer Journal*. Advance online publication. <https://doi.org/10.1093/comjnl/bxad118>.
12. Navin, K. S., Nehemiah, H. K., Jane, Y. N., & Saroji, H. V. (2023). A classification framework using filter-wrapper based feature selection approach for the diagnosis of congenital heart failure. *Journal of Intelligent & Fuzzy Systems*, 44 (4), 6183–6218.
13. Ruzafa-Alcázar, P., Fernández-Saura, P., Marmol-Campos, E., González-Vidal, A., Hernández-Ramos, J. L., Bernal-Bernabe, J., & Skarmeta, A. F. (2021). Intrusion detection based on privacy-preserving federated learning for the industrial IoT. *IEEE Transactions on Industrial Informatics*, 19 (2), 1145–1154.
14. Ullah, I., & Mahmoud, Q. H. (2022). Design and development of RNN anomaly detection model for IoT networks. *IEEE Access*, 10, 62722–62750.
15. Ullah, I., & Mahmoud, Q. H. (2022). An anomaly detection model for IoT networks based on flow and flag features using a feed-forward neural network. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)* (pp. 363–368). IEEE.
16. Saba, T., Rehman, A., Sadad, T., Kolivand, H., & Bahaj, S. A. (2022). Anomaly-based intrusion detection system for IoT networks through deep learning model. *Computers and Electrical Engineering*, 99, 107810.
17. Nascita, A., Cerasuolo, F., Di Monda, D., Aberia Garcia, J. T., Montieri, A., & Pescapé, A. (2022). Machine and deep learning approaches for IoT attack classification. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (pp. 1–6). IEEE.
18. Al Sawafi, Y., Touzene, A., & Hedjam, R. (2023). Hybrid deep learning-based intrusion detection system for RPL IoT networks. *Journal of Sensor and Actuator Networks*, 12 (2), 21.

19. Jithish, J., Alangot, B., Mahalingam, N., & Yeo, K. S. (2023). Distributed anomaly detection in smart grids: A federated learning-based approach. *IEEE Access*, 11, 7157–7179.
20. Kao, M.-T., Sung, D.-Y., Kao, S.-J., & Chang, F.-M. (2022). A novel two-stage deep learning structure for network flow anomaly detection. *Electronics*, 11 (10), 1531.
21. Rejito, J., Stiawan, D., Alshaflut, A., & Budiarto, R. (2024). Machine learning-based anomaly detection for smart home networks under adversarial attack. *Computer Science and Information Technologies*, (2), 122–129.
22. Balega, M., Farag, W., Wu, X.-W., Ezekiel, S., & Good, Z. (2024). Enhancing IoT security: Optimizing anomaly detection through machine learning. *Electronics*, 13 (11), 2148.
23. Nascita, A., Carillo, R., Giampetraglia, F., Iacono, A., Persico, V., & Pescap'e, A. (2024). Interpretability and complexity reduction in IoT network anomaly detection via XAI. In *2024 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops (ICASSPW)* (pp. 325–329). IEEE.
24. Nguyen, Q. H., Hore, S., Shah, A., Le, T., & Bastian, N. D. (2025). FedNIDS: A federated learning framework for Packet-based network intrusion detection system. *Digital Threats: Research and Practice*, 6(1), 1-23.
25. EA, M. A. (2023). A novel paradigm for IoT security: ResNet-GRU model revolutionizes botnet attack detection. *International Journal of Advanced Computer Science & Applications*, 14 (12).