Proceedings of CONF-SEML 2026 Symposium: Computational Analysis and Modeling in Complex Intelligent Syste (CONF-SEML 2026)
DOI: 10.54254/2755-2721/2026.GU34715

Hybrid Framework Using Deep Learning for Botnet Detection in Containerized Environment

Weijia Zhang

Faculty of Science and Technology, Beijing Normal - Hong Kong Baptist University, Zhuhai, China
Justinwj8343@hotmail.com

Abstract. An escalation in sophisticated, encrypted botnet attacks has accompanied the rapid expansion of the Internet of Things (IoT) ecosystem. Conventional security measures are increasingly ineffective due to the rise of encrypted communication channels, while standard machine learning and deep learning models are computationally expensive for local edge deployment. This paper first proposes a hybrid, dual-branch deep learning framework designed for high-fidelity botnet detection within containerized edge environments. Experimental evaluations are then conducted using a curated dataset derived from IoT-23 and tested within a Docker-based simulation. The proposed framework in the resource-constrained setting can achieve high accuracy and a low False Positive Rate while maintaining superior computational efficiency and outperforming traditional baseline classifiers. These findings demonstrate strong capabilities for robust detection at the network edge with negligible memory overhead, offering a scalable solution for modern IoT infrastructures. Finally, the paper provides a critical assessment of the framework's limitations and future research directions of botnet detection in edge deployment scenarios.

Keywords: Botnet, Network Traffic Detection, Machine Learning, Deep Learning, Docker

1. Introduction

The Internet of Things (IoT) is projected to connect over 29 billion devices by 2030, a rapid proliferation that is transforming healthcare efficiency, manufacturing capabilities, and the development of smart cities [1]. However, this growth has fundamentally changed the threat landscape. The inherent nature of IoT devices, with limited security investment, creates a vulnerable attack surface for botmasters [2]. Consequently, these devices have become primary targets for sophisticated malware botnets such as Mirai, Torii, and Hakai.

A botnet is foundational infrastructure for malicious network activity, comprising a network of malware-infected hosts—or "bots"—usually controlled by a Command and Control (C&C) server. Modern botnets have evolved to adopt highly resilient architectures that eliminate single points of failure and bypass traditional detection methods through encrypted communication channels [2]. Recent data indicates that approximately 97% of the world's top websites now use HTTPS, and nearly 91.5% of malware detections involve threats arriving over encrypted connections [3]. The primary challenge for modern security systems is performing "non-decryption-based" detection. Traditional Deep Packet Inspection (DPI) is often powerless against encrypted traffic, and

DOI: 10.54254/2755-2721/2026.GU34715

decrypting data poses significant risks to user privacy [3]. While machine learning (ML) and deep learning (DL) models offer a promising alternative, they can suffer from performance deterioration due to changes in data distribution and limited generalizability across different network environments [1]. Furthermore, research suggests that lightweight, high-fidelity detection systems can be deployed on network edge devices to address the early stages of a botnet's life cycle, such as infection and propagation [4].

This paper proposes a hybrid dual-branch framework that treats network traffic as a multi-dimensional signal. By leveraging a Multi-Layer Perceptron (MLP) branch to capture dynamic behavioral statistical features and a Convolutional Neural Network (CNN) branch to process structural spatial features derived from raw packet bytes, the framework enables comprehensive analysis of connection metadata [5].To ensure practical applicability, the model undergoes dynamic range quantization via TensorFlow Lite (TFLite) for deployment in a resource-constrained containerization environment.

The remainder of this paper is organized as follows: Section 2 details the experiment design, including dataset collection from the IoT-23 dataset by Gracia et al., feature extraction methodologies, and the hardware constraints of the Docker-based simulation environment [6].

Section 3 presents a comparative analysis of the results, evaluating the proposed framework against baseline models in terms of classification performance and computational resource trade-offs. Finally, Section 4 concludes the paper and discusses limitations and future work.

2. Experiment design

2.1. Hybrid deep learning framework for botnet traffic detection

In this section, we propose a dual-branch malicious botnet detection framework comprising two deep learning algorithms. Two sets of features extracted from a curated dataset will be fed into their respective algorithms to improve training and testing results. Statistical features will be selected for the MLP branch that will capture dynamic behavioral network traffic patterns. Spatial features will be selected for the CNN branch that will process image features derived from raw packet bytes to understand structural patterns. Then, the framework will combine the two outputs of each branch through a fusion layer to perform the final prediction. Finally, the model will undergo dynamic range quantization using TensorFlow Lite (TFLite) to enable performance evaluation in resource-constrained environments.

2.2. Dataset collection

This experiment utilizes specific scenarios from the IoT-23 dataset, curated by the Stratosphere Laboratory, to build a smaller dataset [6]. Each scenario in this project consists of a logged, LABELED file containing network traffic flows and a matching PCAP file. A LABELED file contains each flow, represented by a set of statistical features derived from packet-level observations, with a label indicating whether a flow is benign or associated with botnet activity. The specific dataset curated for this experiment includes 7 scenarios, of which 6 involve malicious botnet sessions, as illustrated in Table 1. Each scenario has its session flows matched to the corresponding packets, creating a dataset that includes both numerical and image statistics, as well as labels.

DOI: 10.54254/2755-2721/2026.GU34715

Table 1. Benign and malicious session composition in the dataset

Malicious scenarios	Benign sessions	Malicious sessions
Mirai	1923	21222
Torii	3193	16
Trojan	4420	6
Hakai	2181	8222
Muhstik	3329	151567
Hide and Seek	451854	539473
Benign Scenarios	Benign sessions	Malicious sessions
CTU-Honeypot-Capture-5-1 (Amazon Echo)	1203	-
Total Samples	Total Benign	Total Malicious
1188609	468103	720506

2.3. Experiment setup and performance evaluation

The experiment is run on an AMD Ryzen 9 7940H CPU at 4GHz with 16.0GB RAM, with the use of scikit-learn and TensorFlow libraries for building the model framework. For model performance evaluation, the following metrics are used: accuracy, precision, recall, True Positive Rate (TPR), False Positive Rate (FPR), f1-score, and AUC-ROC. These metrics provide a more comprehensive analysis of classification performance. They are defined as follows:

Accuracy: proportion of correctly classified samples.

Precision: ratio of true positives to predicted positives.

Recall/True Positive Rate (TPR): ratio of true positives to actual positives.

False Positive Rate (FPR): ratio of false positives to actual negatives

F1-score: harmonic mean of precision and recall.

ROC-AUC: area under the Receiver Operating Characteristic curve.

To evaluate the performance of the proposed hybrid framework in an IoT deployment scenario, we used a Docker containerized simulation environment. We employed containerization to enforce hardware constraints in the test environment, in the absence of physical edge hardware, ensuring the original workstation's higher performance does not inflate results. The simulation is designed to represent a mid-range IoT Gateway with 1 vCPU and a RAM limit of 2GB. The experiment will train the model on the original workstation using the dataset and test its performance in a restricted environment.

In addition to classification performance, we will also measure resource usage, specifically the average inference latency per sample, the maximum throughput (samples per second), the estimated GFLOPS/sec, and peak memory (RAM) overhead in the containerized environment. For the quantized model proposed by the framework, computational throughput is measured in effective GFLOPS/sec, representing the equivalent floating-point operations performed via optimized 8-bit integer inference.

DOI: 10.54254/2755-2721/2026.GU34715

2.4. Feature extraction and data preprocessing

The curated dataset for this experiment will be fed into the preprocessing pipeline to produce two feature sets: a statistical set and a spatial set, designed to reflect the characteristics of the deep-learning model in our framework.

(1) Statistical set: We extract 11 features that define a flow's temporal and volumetric behavior. This includes the Mean Inter-Arrival Time (IAT) to detect the rhythmic heartbeat of C2 polling and the Mean Packet Length to identify specific payload signatures of DDoS or exfiltration activities. The 11 specific features below are chosen for their representativeness of the flow:

(a) Mean packet length
(b) Std dev of packet lengths
(c) Mean inter-arrival time
(d) Std dev of inter-arrival times
(e) Max packet length
(f) Min packet length
(g) Mean of the last 3 packet lengths
(h) Original packet number (number of packets sent by originator).
(i) Original IP bytes (total IP bytes sent by originator).
(j) Duration (connection duration in seconds).
(k) Response packets (number of packets sent by responder).

(2) Spatial set: We extract the first 196 bytes of the IP payload. This specific window is chosen because it encompasses the TLS/SSL Handshake (Client Hello, Cipher Suite list, and Handshake Extensions). These bytes are reshaped into a 14x14 grid, which allows the CNN to learn and identify visual cues in the byte distribution that correspond to specific botnet encryption libraries.

Next, we split the entire feature dataset into training and test sets before preprocessing. We implement under-sampling on the training sets to prevent the model from biasing toward the majority class (Malicious sessions). By balancing the training set, we ensure the model learns to identify botnet features with the same weight as benign features. Furthermore, numeric data from the statistical set is normalized to have a mean of zero and unit variance. In contrast, the 14x14 spital data is normalized by a factor of 255.0 to scale pixel intensity between [0, 1].

2.4.1. Statistical features with MLP model

The MLP branch processes the 11 statistical features using a feed-forward neural network designed for 1D statistical data. It consists of two dense layers (64 and 32 neurons). We use the ReLU (Rectified Linear Unit) activation function because it mitigates the vanishing gradient problem, enabling the model to learn complex non-linear relationships in traffic metadata more efficiently than sigmoid functions.

2.4.2. Spatial features with CNN model

The 2D CNN branch processes the grayscale 14x14x1 structural image as input. It uses a 3x3 convolutional kernel to find spatial correlations between handshake bytes and outputs a 1D vector of 1568 elements after the feature map is normalized. This branch focuses on visual anomalies in botnet behavior observed in network traffic.

DOI: 10.54254/2755-2721/2026.GU34715

2.5. Comparative analysis: baseline models

The hybrid model is evaluated against five other models to validate the performance of the dual-branch deep learning framework. The baseline models are trained and evaluated using the same dataset splits and feature sets in identical environments to ensure fair comparisons. The 5 classifiers are listed as follows:

(1) Manual TensorFlow Logistic Regression
(2) Manual TensorFlow Logistic Regression with Hidden Layer
(3) Keras Artificial Neural Network (ANN)
(4) Decision Tree
(5) Random Forest

3. Experiment result and evaluation

As mentioned in section 2.1, the two algorithms in the dual branch framework produce predictions based on their characteristic input feature sets, which are then fed to a fusion layer for the final prediction. Next, the model will undergo quantization and evaluation in the restrained containerized environment. Based on the experimental results, the framework achieves 99.91% accuracy and 99.99% ROC-AUC, as detailed in Table 2, with the lowest false-positive rate, an important factor in operational environments. Baseline models such as Logistic Regression demonstrate strong recall but relatively high false-positive rates. While tree-based and neural models achieve stronger overall performance, the proposed framework demonstrates the need for combined approaches in botnet detection to achieve superior results with low FPR.

Table 2. Comparative classification performance results

Models	Accuracy	F1	Recall (TPR)	Precision	ROC-AUC	FPR
Logistic Regression	0.9379	0.9508	0.9891	0.9153	0.9126	0.1409
Hidden Layer LR	0.9428	0.9548	0.9974	0.9158	0.9282	0.1412
Keras ANN	0.9606	0.9685	0.9993	0.9396	0.9531	0.0988
Decision Tree	0.9620	0.9696	0.9993	0.9416	0.9600	0.0954
Random Forest	0.9595	0.9675	0.9947	0.9418	0.9572	0.0947
Proposed Framework	0.9991	0.9992	0.9996	0.9988	0.9999	0.0018

3.1. Resource trade-offs

In terms of resource consumption in the Docker containerized environment, as shown in Table 3, while Random Forest and ANN models offer strong detection performance, the increased RAM usage could be challenging in constrained scenarios. Decision Trees provide a favorable balance between accuracy and efficiency. In contrast, the proposed framework offers significant advantages in average latency per-sample prediction and maximum throughput, resulting in higher effective GFlops/sec.

Table 3. Comparative resource usage results in a Docker containerized environment

Models	Avg Latency (ms/sample)	Max Throughput (samples/sec)	Estimated Gflops/sec	Peak RAM Overhead (MB)
Logistic Regression	0.1474	6784.43	0.000312	0.00

DOI: 10.54254/2755-2721/2026.GU34715

Table 3. (continued)

Hidden Layer LR	0.2149	4653.34	0.000484	0.00
Keras ANN	1.3942	717.26	0.000475	0.62
Decision Tree	0.0821	12174.21	N/A	0.12
Random Forest	1.5112	661.74	N/A	0.12
Proposed Framework	0.0080	125013.72	40.420436	

4. Conclusion

This paper designed and evaluated a hybrid, dual-branch deep learning framework for the detection of malicious botnet traffic in a resource-constrained containerized environment. By integrating a MLP branch to analyze statistical patterns and a CNN branch to identify spatial cues from packet bytes, the proposed framework captures the multi-dimensional nature of modern botnet threats under TFLite's quantization techniques to ensure the model's practical applicability for IoT devices.

The experimental results demonstrate that the proposed framework significantly outperforms traditional baseline models, such as Random Forest and standard Artificial Neural Networks. The system achieved 99.91% accuracy and a low False Positive Rate (FPR) of 0.0018. Crucially, in a Docker-based simulation of a mid-range IoT gateway (restricted to 1 vCPU and 2GB RAM), the framework maintained superior efficiency, delivering an average inference latency of 0.0080 ms per sample and a maximum throughput of 125,013.72 samples per second. These results validate the framework's ability to provide robust security at the network edge with negligible memory overhead.

Despite promising results, several major limitations remain. First, the nature of simulation using a Docker container means the performance is not fully representative of physical ARM-based or specialized edge hardware. Second, the training process uses an under-sampling strategy to balance the dataset, which may not reflect real-world network environments where malicious sessions are outnumbered by benign traffic. Thirdly, the current framework is optimized for binary classification (malicious vs. benign) and does not differentiate between specific botnet families in real time.

Finally, the current experiment is not extensive enough, with a limited selection of comparative baseline classifiers, which may inflate the framework's performance results.

Future research will focus on expanding the framework to support multi-class classification, enabling the system to identify specific types of botnets and support more tailored incident responses. In order to assess real-time botnet detection and classification outcomes as well as computing resource utilisation, we also intend to update the framework to collaborate with current network monitoring software. Finally, it plan to deploy the quantized model on physical IoT devices, such as Raspberry Pi or NVIDIA Jetson modules, to further validate its performance and power consumption in a live, physical edge computing scenario.

References

[1] Al-Fawa'reh, M., Abu-Khalaf, J., Szewczyk, P., & Kang, J. J., "MalBoT-DRL: Malware botnet detection using deep reinforcement learning in IoT networks, " IEEE Internet of Things Journal, volume 11, issue 6, p. 9610–9629, 2024.

[2] Vu, S. N. T., Stege, M., El-Habr, P. I., Bang, J., & Dragoni, N., "A survey on botnets: Incentives, evolution, detection and current trends, " Future Internet, volume 13, issue 8, p. 198, 2021.

[3] Wang, Z., & Thing, V. L. L., "Feature mining for encrypted malicious traffic detection with deep learning and other machine learning algorithms, " Computers & Security, volume 128, p. 103143, 2023.

DOI: 10.54254/2755-2721/2026.GU34715

[4] Nadeem, M. W., Goh, H. G., Aun, Y., & Ponnusamy, V., "Detecting and mitigating botnet attacks in software-defined networks using deep learning techniques, " IEEE Access, volume 11, p. 49153–49171, 2023.
[5] Hong, Y., Li, Q., Yang, Y., & Shen, M., "Graph-based encrypted malicious traffic detection with hybrid analysis of multi-view features, " Information Sciences, volume 644, issue 119229, 2023.
[6] Garcia, S., Parmisano, A., & Erquiaga, M. J., "IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set], " Zenodo, 2020.