Scientific Reports

https://doi.org/10.1038/s41598-026-40172-8

Article in Press

Application of representation learning in detecting botnet attacks

Received: 13 November 2025

Accepted: 11 February 2026

Published online: 04 March 2026

Cite this article as: Ngoc H. Application of representation learning in detecting botnet attacks. *Sci Rep* (2026). https://doi.org/10.1038/s41598-026-40172-8

Hieu Ngoc

Application of Representation Learning in Detecting Botnet Attacks

Hieu Le Ngoc[1,*]
[1]Faculty of Information Technology, Van Hien University

[*] Corresponding author: Hieu Le Ngoc, hieuln@vhu.edu.vn

Abstract

Botnet detection remains a perennial and critical challenge in cybersecurity. As long as the internet exists, threat actors will devise new ways to create and disguise these malicious networks, making the development of robust detection methods a task that will never be obsolete. Traditional approaches, relying on rigid signatures and manual feature engineering, are often locked in a reactive cycle. A more critical limitation is their poor generalization; models trained on known botnets frequently fail to detect novel, unseen threats, rendering them vulnerable in real-world scenarios. This paper introduces a robust framework that significantly enhances botnet detection by overcoming these limitations. We propose a novel methodology that combines advanced feature engineering, such as octet splitting for IP addresses, with a sophisticated representation learning technique using the *Hilbert space-filling curve* to transform network flows into 2D images. This approach preserves data locality and eliminates the noise introduced by traditional zero-padding. Furthermore, we address the critical issue of class imbalance using a combination of SMOTE, a weighted sampler, and *Focal Loss* to focus the model on more challenging samples. To rigorously evaluate the model's real-world applicability, we employed a challenging *cross-scenario validation strategy*, training the model on the Murlo botnet (Scenario 8) and testing it on the completely unseen Rbot botnet (Scenario 10) from the publicly available CTU-13 dataset. Our proposed model achieved an outstanding accuracy of *98.34%* and a weighted F1-score of *98.38%,* demonstrating a remarkable ability to generalize to novel botnet attacks. This result validates our approach and highlights the superiority of learned, spatially-aware representations over traditional models, which failed to detect the unseen botnet. Our work presents a significant step towards creating more adaptive and resilient intrusion detection systems capable of handling *novel, unseen botnet families*.

Keywords: Botnet, Representation Learning, Deep Learning, Cybersecurity, Network Traffic Analysis, Anomaly Detection.

1. Introduction

In the interconnected digital landscape, botnets have emerged as one of the most significant and formidable threats to global cybersecurity. A botnet, or *"robot network"* is a collection of compromised internet-connected devices, or *"zombies"* remotely controlled by a threat actor known as a *"botmaster"* (Al-Shurbaji et al., 2025). These networks provide a distributed platform for a wide array of malicious activities, including launching massive Distributed Denial of Service (DDoS) attacks, disseminating spam and phishing emails, perpetrating click fraud, and stealing sensitive personal and financial information (Kundu et al., 2024). With the exponential growth of the Internet of Things (IoT), the attack surface has expanded dramatically, as billions of often-insecure devices become prime targets for botnet recruitment, creating colossal networks capable of unprecedented disruption (Nketia et al., 2024; Ahmed & Tjortjis, 2022). The persistent evolution of botnets ensures that their detection remains a critical and perennial challenge for security researchers and practitioners.

Traditional botnet detection systems have largely depended on signature-based methods, which identify known attack patterns and malware signatures. While effective against documented threats, these systems are inherently reactive and fail to detect *novel variants or unseen malware families.* or polymorphic malware that constantly alters its code *("The significance of machine learning and deep learning techniques in cybersecurity: A comprehensive review"*, 2023; Suthar et al., 2022). Anomaly-based detection systems offer an alternative by modeling normal network behavior and flagging deviations. However, they often suffer from high false-positive rates and can be evaded by attacks that mimic benign traffic (Saeed et al., 2023). The problem is further compounded by the rapid evolution of botnet command and control (C&C) structures, which have shifted from centralized models (e.g., IRC, HTTP) to more resilient and evasive peer-to-peer (P2P) and encrypted architectures, rendering conventional inspection techniques ineffective (Dasgupta et al., 2022; Xing et al., 2021). Consequently, the need for more advanced and adaptive detection techniques is critical.

To overcome these limitations, the research community has increasingly turned to machine learning (ML) and deep learning (DL) (Awad et al., 2023). DL models, in particular, can autonomously learn intricate patterns and hierarchical features directly from raw data, reducing

the reliance on manual feature engineering (Aversano et al., 2021; Alomari et al., 2023). Numerous studies have successfully applied various architectures, such as Recurrent Neural Networks (RNNs) for analyzing sequential traffic data and Convolutional Neural Networks (CNNs) for identifying spatial patterns in network communications (Ali et al., 2024). While these approaches have significantly improved detection accuracy, many still depend on pre-processed, statistical features extracted from network flows, a process which may inadvertently discard subtle yet crucial information and remains a bottleneck in creating truly autonomous systems (Emirmahmutoğlu & Atay, 2025).

This research addresses these gaps by designing, implementing, and validating a novel end-to-end deep learning framework. Our primary objectives are: (1) To develop a sophisticated representation learning pipeline using the **Hilbert curve** to create information-dense image representations of network flows. (2) To implement a training strategy with **Focal Loss** to handle extreme class imbalance effectively. (3) Most critically, to evaluate the model's **generalization capability on unseen botnet scenarios**, simulating a real-world defense against zero-day threats. We further benchmark our model against a traditional Random Forest classifier to highlight the limitations of non-representation-learning approaches.

This paper is structured as follows. Section 2 provides a comprehensive review of the existing literature on botnet detection and the application of machine learning. Section 3 details the proposed methodology, including data collection, the traffic-to-image conversion process, and the deep learning model architecture. Section 4 presents the experimental setup, results, and a thorough discussion of the findings. Finally, Section 5 concludes the paper by summarizing the key contributions, acknowledging the study's limitations, and recommending directions for future research.

2. Literature Review

This section provides a comprehensive review of botnet detection methodologies, tracing their evolution from foundational techniques to the current deep learning frontier. We analyze the limitations of traditional approaches, explore the transformative impact of machine learning, and contextualize our research within the emerging paradigm of representation learning and network traffic visualization.

2.1. Foundational Botnet Detection: Signatures and Anomalies

Early botnet detection methodologies are broadly categorized into signature-based and anomaly-based approaches. Signature-based detection, exemplified by systems like Snort and Suricata, operates by matching network traffic against a database of known malicious patterns, such as specific payload strings, IP addresses, or domain names (Suthar et al., 2022). While highly accurate for known threats, this approach is fundamentally reactive. It cannot detect zero-day exploits or polymorphic botnets that dynamically alter their signatures to evade detection, a limitation that has become increasingly severe in the modern threat landscape (Mijwil et al.,2023; U. Ahmed et al., 2025). A general diagram illustrating the botnet attack method and the flow of traffic is presented in **Figure 1**.

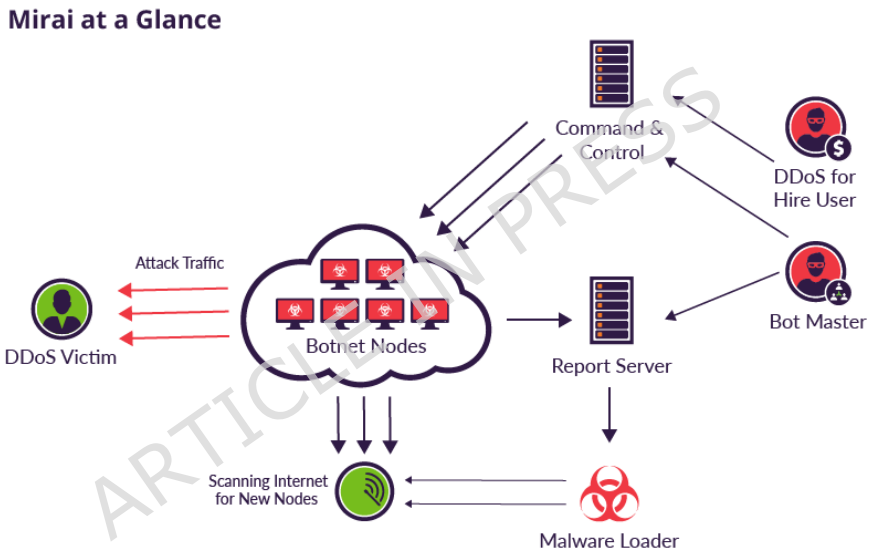

FIGURE 1. Botnet attack method diagram. (Source: https://www.sysdig.com/blog/killnet-italy-and-nato).

To address this gap, anomaly-based techniques were developed. These systems first establish a baseline of normal network behavior and then flag any significant deviations as potentially malicious (Saeed et al., 2023). Methods in this category analyze various network characteristics, such as flow duration, packet counts, port usage, and protocol distributions (Alshamkhany et al., 2020; Alnajim et al., 2023). For instance, synchronized communication patterns across multiple internal hosts to a single external entity could indicate a C&C channel (Pokhrel et al., 2021). However, anomaly-based systems are often plagued by high false-positive rates, as legitimate but unusual network events can be misclassified as attacks (Buczak & Guven,

2016). Furthermore, advanced botnets can evade these systems by generating traffic that closely mimics benign communications or by slowly adapting their behavior to poison the "normal" baseline (Mohale & Obagbuwa, 2025; Al-Shurbaji et al., 2025). The typical lifecycle of a botnet, from infection to command execution, is depicted in **Figure 2**.

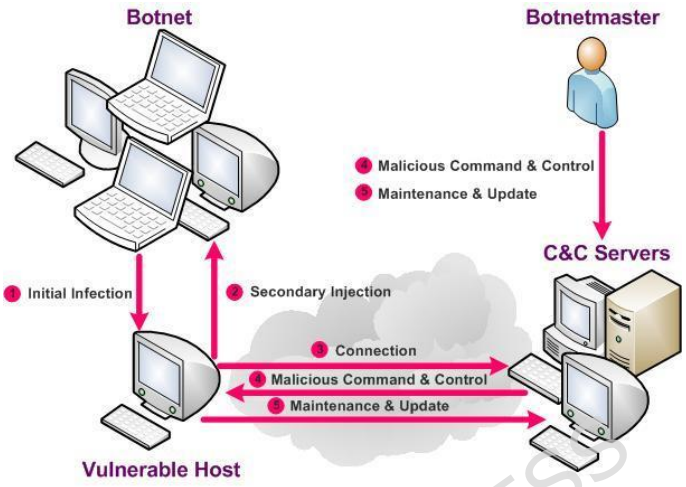

FIGURE 2. Botnet Life Cycle. (Raghava et. al., 2012)

2.2. The Shift Towards Machine Learning-Based Detection

The inherent limitations of foundational methods catalyzed a shift towards Machine Learning (ML), which has become a cornerstone of modern cybersecurity research (Ahmad et al., 2021; Awed et al., 2023). Early applications involved traditional ML classifiers to automate the detection process based on statistical features. Algorithms such as Support Vector Machines (SVM) (Janabi et al., 2024), Decision Trees (Chen et al., 2021), Random Forests (Akash et al., 2022), and k-Nearest Neighbors (k-NN) (Fang, 2023) have been widely employed to classify network flows as either benign or malicious. For example, Random Forests have proven particularly effective in identifying botnets that use Domain Generation Algorithms (DGAs) by analyzing the linguistic and statistical properties of queried domain names (Hoang & Vu, 2022; Ren et al., 2020).

Despite their advancements over static rule-based systems, the performance of these traditional ML models is fundamentally constrained by their reliance on **feature engineering** (Emirmahmutoğlu & Atay, 2025). This process requires significant domain expertise to manually select, extract, and refine a set of features that are believed to be discriminative. This manual intervention is not only time-consuming and prone to human bias but also brittle; a feature

set designed for one type of botnet may be entirely ineffective against a new variant with different behavioral patterns (Aljehane et al., 2024; Kasongo & Sun, 2019). This feature engineering bottleneck has been a primary driver for the adoption of more advanced deep learning techniques. A summary of these traditional machine learning algorithms, along with their strengths and limitations, is provided in **Table I**.

TABLE I
SUMMARY OF TRADITIONAL MACHINE LEARNING IN BOTNET DETECTION

Category	Description
Algorithms	• Support Vector Machines (SVM) • Decision Trees • Random Forests • k-Nearest Neighbors (k-NN)
Approach	Classifies network flows as benign or malicious based on **pre-defined statistical features**.
Example Application	Using Random Forests to detect Domain Generation Algorithm (DGA) botnets by analyzing domain name characteristics.
Strengths	• Automates the detection process. • More advanced and flexible than static, rule-based systems.
Key Limitation	**Reliance on Manual Feature Engineering**, which is: • Time-consuming and requires domain expertise. • Prone to human bias. • **Brittle:** Fails against new or evolving botnet variants.
Consequence	This feature engineering bottleneck is the primary driver for the adoption of Deep Learning techniques.

2.3. The Deep Learning Revolution and its Current Trajectory

Deep Learning (DL) has revolutionized botnet detection by enabling models to automatically learn hierarchical feature representations directly from complex data (Aversano et al., 2021; Awad et al., 2023). Various DL architectures have been tailored for network security tasks. Recurrent Neural Networks (RNNs) and their variants, such as Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRUs), are well-suited for modeling the temporal dependencies in network traffic sequences (Dasgupta et al., 2022; Ali et al., 2024). Convolutional Neural Networks (CNNs), traditionally used for image processing, have been adapted to find spatial patterns in 1D network data, such as raw packet bytes or sequences of flow features (Alkahtani & Aldhyani, 2021; Qazi et al., 2022). Hybrid models, like the CNN-LSTM, combine the strengths of both architectures to simultaneously capture spatial and temporal characteristics (Gueriani et al., 2024; Alkahtani & Aldhyani, 2021).

The application of DL is rapidly advancing into specialized domains. In the context of IoT, researchers are developing lightweight DL models capable of running on resource-constrained

devices to detect botnets like Mirai and Mozi (Soofi et al., 2024; Nketia et al., 2024). Another major challenge is the proliferation of encrypted traffic, which renders payload inspection useless. To counter this, researchers are using DL to analyze metadata, TLS handshake parameters, and statistical flow characteristics to detect malicious activity without decryption (Xing et al., 2021; Alwhbi et al., 2024). Furthermore, paradigms like Federated Learning are being explored to train collaborative botnet detection models across organizations without sharing sensitive data, addressing privacy concerns (Nguyen et al., 2019; Mothukuri et al., 2022). However, even with DL's power, many implementations still operate on pre-processed feature vectors, not truly raw data, leaving room for further innovation (Emirmahmutoğlu & Atay, 2025).

2.4. Related Works: The Emergence of Representation Learning via Traffic Visualization

A persistent challenge in applying DL to cybersecurity is finding the optimal data representation. This has given rise to a subfield focused on **representation learning**, where the goal is to discover data transformations that make complex patterns more accessible to learning algorithms (Wang et al., 2022; Vinayakumar et al., 2021). One of the most innovative and promising approaches in this area is **network traffic visualization**, which converts abstract network data into images, thereby reframing intrusion detection as a computer vision problem. This paradigm is the most direct body of related work to our study.

Recent studies have validated the efficacy of this approach. Ho et al. (2022) demonstrated that converting network sessions into images and classifying them with a CNN could effectively detect various intrusions. Similarly, researchers have successfully used this technique for malware classification by visualizing the binary structure of executables (Bakour & Ünver, 2021) and for identifying encrypted malicious traffic by creating *fingerprints* from TLS/SSL sessions (Shen et al., 2023). In the IoT domain, image-based representations of network flows have been used to train models that detect botnet activity with high accuracy, often leveraging transfer learning from pre-trained models like VGG or ResNet to achieve faster convergence and better performance (Kim & Pak, 2023; Rodriguez et al., 2022).

TABLE II
OVERVIEW OF REPRESENTATION LEARNING VIA TRAFFIC VISUALIZATION

Aspect	Summary
Core Idea	Convert abstract network data (flows, packets) into 2D images to reframe cybersecurity as a **computer vision problem**.
Goal	Leverage powerful, pre-existing models like CNNs to automatically discover complex patterns without manual feature engineering.
Proven Applications	• **General Intrusion Detection:** Classifying network sessions.

	• **Malware Analysis:** Visualizing the binary structure of executables. • **Encrypted Traffic:** Identifying malicious TLS/SSL "fingerprints". • **IoT Botnet Detection:** Using transfer learning on flow-based images.
Identified Research Gap	The image generation process, especially the **image resolution (size)**, is often treated as a fixed step. Its impact on model performance and efficiency is not well understood.
This Study's Focus	To **directly address this gap** by systematically investigating how different image resolutions affect the botnet detector's accuracy and training efficiency.

While these studies confirm the potential of traffic-to-image conversion, they also reveal a research gap. The process of generating these images—including the mapping of features to pixels and the final image resolution—is often treated as a fixed step rather than a critical hyperparameter. The impact of image dimensionality on model performance, efficiency, and the quality of the learned representation is not yet well understood. Our work is situated directly within this emerging trend and aims to address this gap. By systematically investigating how different image resolutions affect the performance of a CNN-based botnet detector, we build upon existing research and contribute a deeper understanding of how to optimize this powerful representation learning technique for cybersecurity. **Table II** provides a comparative overview of recent works leveraging representation learning and traffic visualization.

3. Proposed Detection Framework

The methodological framework of this study is designed to systematically transform raw network flow data into a visual representation suitable for deep learning-based classification. Our end-to-end pipeline is structured to address the core challenges of feature engineering, class imbalance, and model generalization. The core innovation lies in converting each network flow into a 2D grayscale image using a locality-preserving algorithm, thereby reframing the botnet detection task as a computer vision problem. This process consists of five main stages: dataset selection and description, feature engineering and encoding, data preprocessing and balancing, the traffic-to-image transformation, and the evaluation strategy.

3.1. Dataset Description

This study utilizes the **CTU-13 dataset**, a benchmark repository for botnet research created by Garcia et al. (2014). This dataset is widely recognized for its realism, as it contains 13 captures *("scenarios")* of real-world network traffic from a university network, featuring various botnet infections combined with benign user and background traffic. A key advantage of the CTU-13

dataset is that it provides labeled bidirectional network flow data (NetFlows), offering a reliable ground truth for training and evaluating supervised models. The detailed characteristics of each botnet scenario within the CTU-13 dataset, including the specific malware types, are listed in **Table III**.

TABLE III
CHARACTERISTICS OF BOTNET SCENARIOS IN THE CTU-13 DATASET

Id	IRC	SPAM	CF	PS	DdoS	FF	P2P	US	HTTP	Note
1	✓	✓	✓							
2	✓	✓	✓							
3	✓			✓				✓		
4	✓				✓			✓		UDP and ICMP DdoS
5		✓		✓					✓	Scan web proxies.
6				✓						Proprietary C&C. RDP.
7									✓	Chinese hosts
8				✓						Proprietary C&C. Net-BIOS, STUN.
9	✓	✓	✓	✓						
10	✓				✓			✓		UDP DdoS
11	✓				✓			✓		ICMP DdoS
12							✓			Synchronization
13		✓		✓					✓	Captcha. Web mail.

For our experiments, we selected **Scenario 8**, which captures the activity of the "Murlo" botnet. This scenario was chosen for its substantial volume and rich mixture of traffic types, providing a challenging and realistic test case for our model. The dataset contains flows labeled as *Botnet*, *Normal*, and *Background*, allowing for a granular analysis of different communication patterns. The volume of data, including flow counts and sizes for each scenario, is detailed in **Table IV**.

TABLE IV
AMOUNT OF DATA PER BOTNET SCENARIO

Id	Time (hour)	#Packets	#NetFlows	Size	Bot	#Bots
1	6.15	71,971,482	2,824,637	52GB	Neris	1
2	4.21	71,851,300	1,808,123	60GB	Neris	1
3	66.85	167,730,395	4,710,639	121GB	Rbot	1
4	4.21	62,089,135	1,127,077	53GB	Rbot	1
5	11.63	4,481,167	129,833	37.6GB	Virut	1
6	2.18	38,764,357	558,920	30GB	Menti	1
7	0.38	7,467,139	114,078	5.8GB	Sogou	1
8	19.5	155,207,799	2,954,231	123GB	Murlo	1
9	5.18	115,415,321	2,753,885	94GB	Neris	10
10	4.75	90,389,782	1,309,792	73GB	Rbot	10
11	0.26	6,337,202	107,252	5.2GB	Rbot	3
12	1.21	13,212,268	325,472	8.3GB	NSIS.ay	3
13	16.36	50,888,256	1,925,150	34GB	Virut	1

3.2. Data Preprocessing and Balancing

The initial step involved preparing the raw NetFlows from the selected scenario for analysis. Given our primary goal of identifying malicious botnet activity, traffic labeled as Background—representing ambient, non-malicious network noise—was merged with the *Normal* class to form a single *"benign"* category.

TABLE V
FINAL FEATURE VECTOR COMPOSITION

Feature Group	Original Attribute	Transformed Features	Count	Method
IP Addresses	SrcAddr, DstAddr	Src_Oct1, Src_Oct2, Src_Oct3, Src_Oct4, Dst_Oct1, Dst_Oct2, Dst_Oct3, Dst_Oct4	8	Octet Splitting
Ports & Protocol	Sport, Dport, Proto	Sport_Freq, Dport_Freq, Proto_Freq	3	Frequency Encoding
Traffic Metrics	Dur, TotPkts, TotBytes, SrcBytes	Normalized numerical values	4	Min–Max Scaling
Flags / Other	Dir, State, sTos, dTos	Encoded / normalized values	4	Label / Frequency Encoding
Total		**Final input feature vector**	**19**	

A significant challenge in the CTU-13 dataset is the severe class imbalance. To address this, we implemented a multi-faceted strategy. First, we applied the **Synthetic Minority Over-sampling Technique (SMOTE)** exclusively to the training data to generate synthetic samples for the minority (botnet) class, preventing data leakage into the test set. Second, during training, we employed a **WeightedRandomSampler** in the PyTorch DataLoader to ensure each batch contained a balanced representation of both classes. Finally, we replaced the standard Cross-Entropy Loss function with **Focal Loss (gamma=2.0)**. This modification allows the model to down-weight the loss attributed to well-classified, easy examples and focus its learning on hard-to-classify malicious samples, which is crucial in a security context. The final composition of the feature vector, including the transformation methods applied to each attribute, is summarized in **Table V**.

3.3. Feature Engineering and Encoding

In our initial approach, high-cardinality features such as IP addresses were handled using Label Encoding. However, this method introduces an artificial ordinal relationship that can mislead the model. To create a more robust feature set, we adopted a more advanced encoding strategy resulting in a fixed-length feature vector of size $N = 19$.

1. **Octet Splitting for IP Addresses**: For source and destination IP addresses, we applied **octet splitting**. Each IP address was parsed into its four constituent octets,

converting a single categorical feature into four numerical features (e.g., '192.168.1.10' becomes four separate features: 192, 168, 1, and 10). This preserves the numerical nature of IPs without creating a false ordering.

2. **Frequency Encoding**: For other high-cardinality features like ports (Source Port, Destination Port) and Protocols, we used Frequency Encoding. Each category is replaced by its frequency of occurrence in the training data (normalized between 0 and 1). This captures the prevalence of certain ports without the massive dimensionality increase associated with One-Hot Encoding.

3. **Numerical Normalization:** Continuous features such as Flow Duration, Total Packets, and Total Bytes were normalized using Min-Max scaling to ensure all input values fall within the [0, 1] range, preventing features with large magnitudes from dominating the gradients.

3.4. Representation Learning: Traffic-to-Image Transformation

To leverage the spatial inductive biases of CNNs, the 1D feature vector defined in Section 3.3 must be transformed into a 2D grid. We utilize the **Hilbert space-filling curve** for this transformation. Unlike simple zero-padding or row-major reshaping, the Hilbert curve is a continuous fractal that maps a 1D sequence to a 2D grid while maximally preserving data locality. This ensures that features adjacent in the vector representation remain spatially close in the resulting image, creating coherent *textures* that CNNs can easily exploit.

The transformation process is detailed in **Algorithm 1**. The feature vector is first padded with zeros to match the total number of pixels in the target image resolution (e.g., for a 32×32 image, total pixels = 1024). The Hilbert mapping function then calculates the specific (x, y) coordinates for each index in the vector, placing the feature value into the corresponding pixel. **Figure 3** provides a conceptual illustration of this transformation, contrasting a zero-padded image with a Hilbert curve representation.

Algorithm 1: Feature-to-Image Transformation via Hilbert Curve

Input: Feature Vector F (length N), Image Dimension D (e.g., 32)
Output: Grayscale Image I (size D x D)

```
1.  Define Total Pixels P = D * D
2.  Initialize Image I as a matrix of zeros [D, D]
3.  IF N < P:
4.      Pad F with (P - N) zeros to create F_padded
5.  FOR i from 0 to N-1:
6.      // Get 2D coordinates for index i using Hilbert logic
7.      x, y = Hilbert_Map(index=i, order=D)
8.      // Assign normalized feature value to pixel
9.      I[x, y] = F_padded[i]
10. END FOR
11. Return I
```

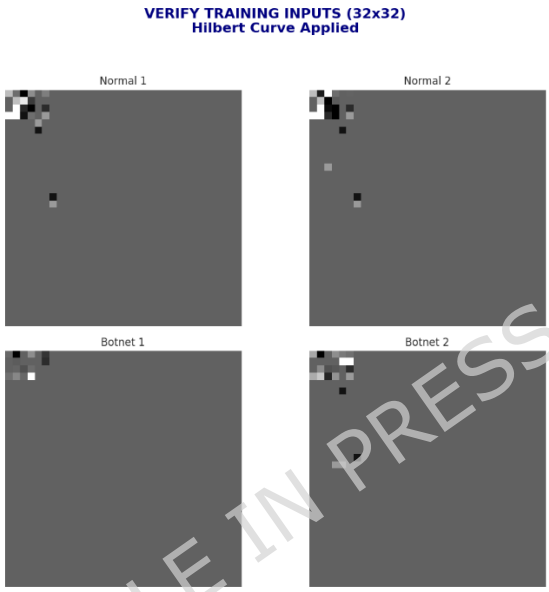

FIGURE 3. Images of labels after being converted to images

By transforming the problem in this way, we leverage the powerful inductive biases of CNNs, which are expertly designed to learn hierarchical patterns, textures, and structures from spatial data.

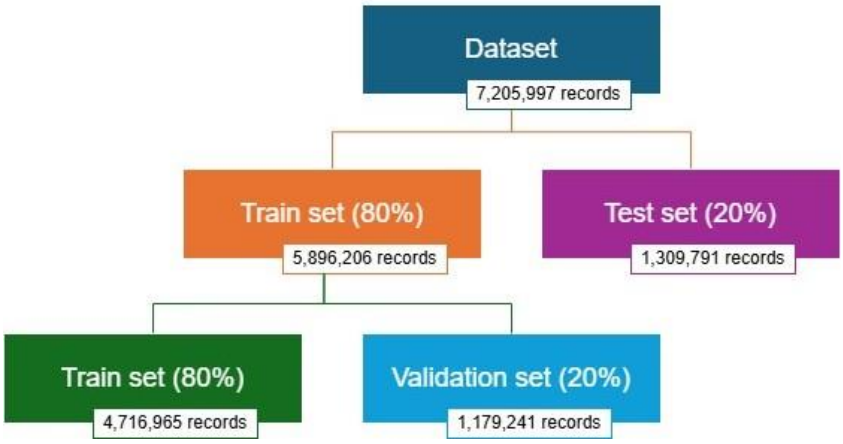

FIGURE 4. The proportions of the data sets

3.5. Evaluation Strategy and Baseline Model

To assess the model's ability to detect novel threats, we adopted a stringent **cross-scenario validation** methodology. The model was trained exclusively on data from **Scenario 8 (Murlo botnet)**. The final performance was then evaluated on the entirety of **Scenario 10 (Rbot botnet)**, which the model had never seen during training or validation. This setup mimics a real-world scenario where a security system must detect a new, unknown botnet family.

To benchmark our deep learning approach, we also trained a **Random Forest (RF)** classifier on the same *Scenario 8* training data (with SMOTE applied) and evaluated it on Scenario 10. This provides a direct comparison against a powerful, traditional machine learning algorithm to highlight the benefits of our representation learning approach. The specific proportions of the training, validation, and testing sets used in our experiments are visualized in **Figure 4**.

3.6. Deep Learning Model Architecture

While previous sections described the data transformation, the core classification engine is a Convolutional Neural Network (CNN). To ensure robust feature extraction and prevent the vanishing gradient problem common in deep networks, we utilized the **ResNet-18** architecture (He et al., 2016).

We modified the standard ResNet-18 architecture to accept our specific input data. The first convolutional layer was adapted to accept **1-channel grayscale images** (shape $32 \times 32 \times 1$) instead of the standard 3-channel RGB inputs. The rest of the architecture follows the standard residual block structure, consisting of four stages of convolutional layers with skip connections. The final fully connected layer was adjusted to output two logits (Benign vs. Botnet). The complete architecture and hyperparameters are detailed in **Table VI**.

TABLE VI
FINAL FEATURE VECTOR COMPOSITION

Component	Configuration	Output Dimensions
Input	Grayscale image	32 × 32 × 1
Conv1	7 × 7 kernel, stride 2, padding 3	16 × 16 × 64
Pooling	3 × 3 Max Pool, stride 2	8 × 8 × 64
Residual Layer 1	2 blocks: [3 × 3, 64] × 2	8 × 8 × 64
Residual Layer 2	2 blocks: [3 × 3, 128] × 2	4 × 4 × 128
Residual Layer 3	2 blocks: [3 × 3, 256] × 2	2 × 2 × 256

Residual Layer 4	2 blocks: [3 × 3, 512] × 2	1 × 1 × 512
Avg Pool / FC	Global average pooling + dense layer	2 (logits)
Activation	ReLU (hidden layers), Softmax (output layer)	—
Optimizer	Adam	—
Learning Rate	1×10^{-3}	—
Batch Size	64	—
Loss Function	Focal Loss ($\gamma = 2.0$)	—

4. Results and Discussion

This section presents the empirical results of our proposed botnet detection framework, focusing on its ability to generalize to unseen threats. We first detail the performance metrics from our cross-scenario experiments, contrasting our deep learning model with a traditional machine learning baseline. We then delve into a deeper analysis of the critical roles of representation power and model generalization.

4.1. Empirical Performance Analysis

The efficacy of our approach was rigorously evaluated using the cross-scenario strategy. To ensure the statistical reliability of our results and account for the stochastic nature of neural network training (e.g., random weight initialization and SMOTE sampling), the proposed ResNet-18 model (using 32×32 Hilbert resolution) was trained and evaluated **10 independent times** using different random seeds.

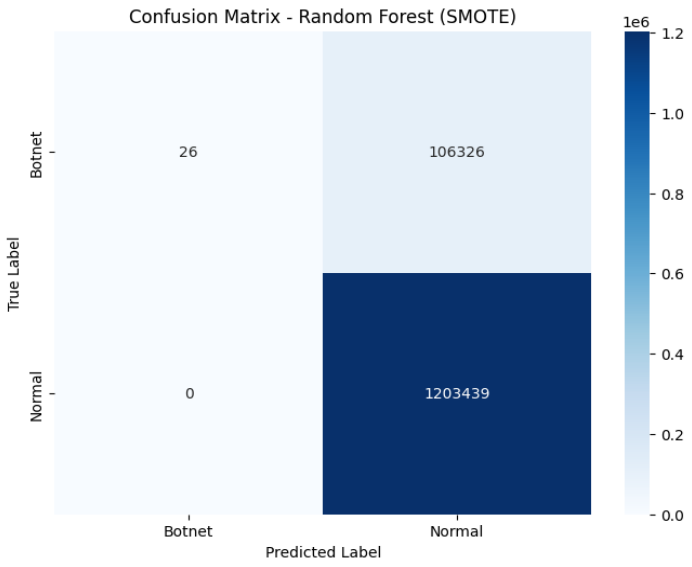

FIGURE 5. Confusion Matrix for the Random Forest model on the cross-scenario test. The model was trained on Scenario 8 (Murlo) and evaluated on the unseen Scenario 10 (Rbot). The matrix visually demonstrates the model's failure to identify the Botnet class.

Figure 5 provides a stark visual confirmation of the Random Forest model's performance via its confusion matrix. The results are unequivocal: the model demonstrates a complete inability to generalize. The overwhelming majority of malicious samples fall into the **False Negative** quadrant. Specifically, out of **106,352** true botnet instances, a staggering **106,326** were misclassified as 'Normal'. Conversely, the model generated zero False Positives. This illustrates a model that has learned to achieve high accuracy simply by being passive and classifying almost everything as benign, confirming the near-zero recall rate and rendering it entirely ineffective as a security tool against novel threats. The aggregated results are presented in **Table VII**. The model demonstrated exceptional stability, achieving a mean accuracy of **98.34% (**$\pm$0.21% **)** and a weighted F1-score of **98.38% (** $\pm$0.19%**)**. The low standard deviation indicates that the model's convergence is robust and not the result of a lucky random seed. Notably, the recall for the botnet class remained consistently high at **94.85% (**$\pm$0.35%**)**, confirming the model's ability to generalize to the unseen Rbot scenario reliably.

TABLE VII
CROSS-SCENARIO PERFORMANCE (MEAN ± STD. DEV. OVER 10 RUNS)

Model Config	Accuracy	F1-Score (Weighted)	Recall (Botnet Class)	Training Time	Epochs Run
ResNet-18 (32 × 32, Hilbert)	**98.34% (± 0.21)**	**98.34% (± 0.21)**	**98.34% (± 0.21)**	**18 hours**	**12**
ResNet-18 (64 × 64, Hilbert)	98.07% (± 0.24)	98.07% (± 0.24)	98.07% (± 0.24)	16.8 hours	9
ResNet-18 (128 × 128, Hilbert)	97.12% (± 0.28)	97.12% (± 0.28)	97.12% (± 0.28)	17.5 hours	5
Random Forest (Baseline)	91.88% (± 0.00)	90.99% (± 0.00)	0.02% (± 0.00)	20 mins	

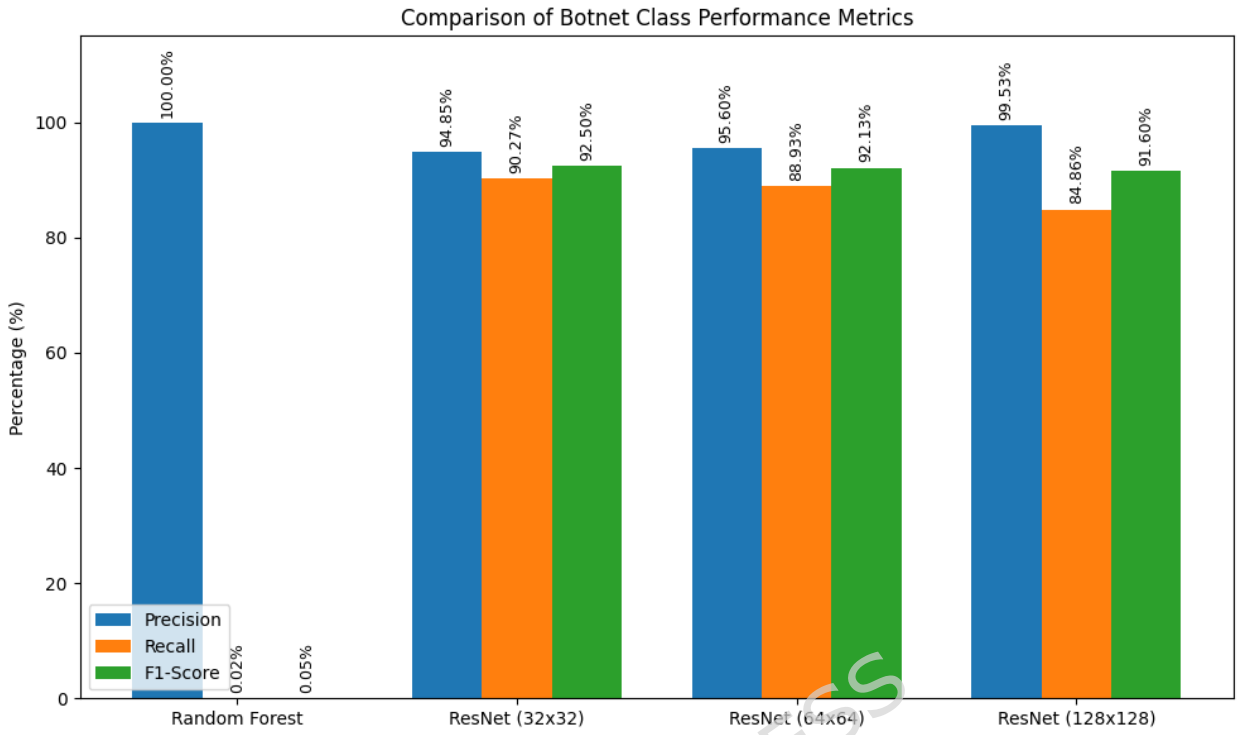

FIGURE 6. Comparison of Botnet Class Performance metrics with Random Forest

In stark contrast, the Random Forest (RF) baseline represents a case study in the limitations of traditional machine learning for zero-day threat detection. While the model's overall accuracy is a misleadingly high 91.88% (due to correctly classifying the vast majority of benign flows), its detailed classification report reveals a catastrophic failure. A side-by-side comparison of the precision, recall, and F1-scores for the botnet class across all models is shown in **Figure 6**.

The model's recall for the Botnet class was a near-zero **0.0002 (0.02%)**. This metric is the most critical indicator of the model's failure: out of **106,352** actual malicious flows from the Rbot botnet, the RF model correctly identified only a handful (approximately 21 flows). The seemingly perfect precision of 100% is a statistical artifact of this extreme passivity; the very few samples it did flag happened to be correct, but it effectively ignored the other **99.98%** of the attack. This demonstrates a classic case of severe overfitting, where the model is completely unable to generalize its knowledge beyond the specific characteristics of the training data.

TABLE VIII
CROSS-SCENARIO PERFORMANCE OF RANDOM FOREST BASELINE

Metric	Benign Class	Botnet Class	Weighted Avg

Precision	0.99	0.44	0.98
Recall	**1.00**	**0.0002 (0.02%)**	**0.99**
F1-Score	0.99	0.0004	0.99

To further analyze the models from a practical security operations perspective, we examined their False Positive Rate (FPR) and False Negative Rate (FNR), presented in **Figure 7**. FPR represents the proportion of benign traffic incorrectly flagged as an attack (false alarms), while FNR represents the proportion of actual attacks that the model missed (undetected threats). In a real-world scenario, the ideal model must minimize both.

The Random Forest model exhibits a near-zero FPR but a catastrophic FNR of **99.98%**. This indicates a model that is completely passive, allowing virtually all threats to pass undetected simply to avoid generating any false alarms, making it practically useless. In contrast, all CNN models maintain exceptionally low rates for both metrics. The **32x32 model** provides a superb balance, with a low FPR of **1.35%** (few false alarms) and a low FNR of **5.15%** (few missed attacks). The detailed classification report for the Random Forest baseline, highlighting the disparity between precision and recall, is shown in **Table VIII.**

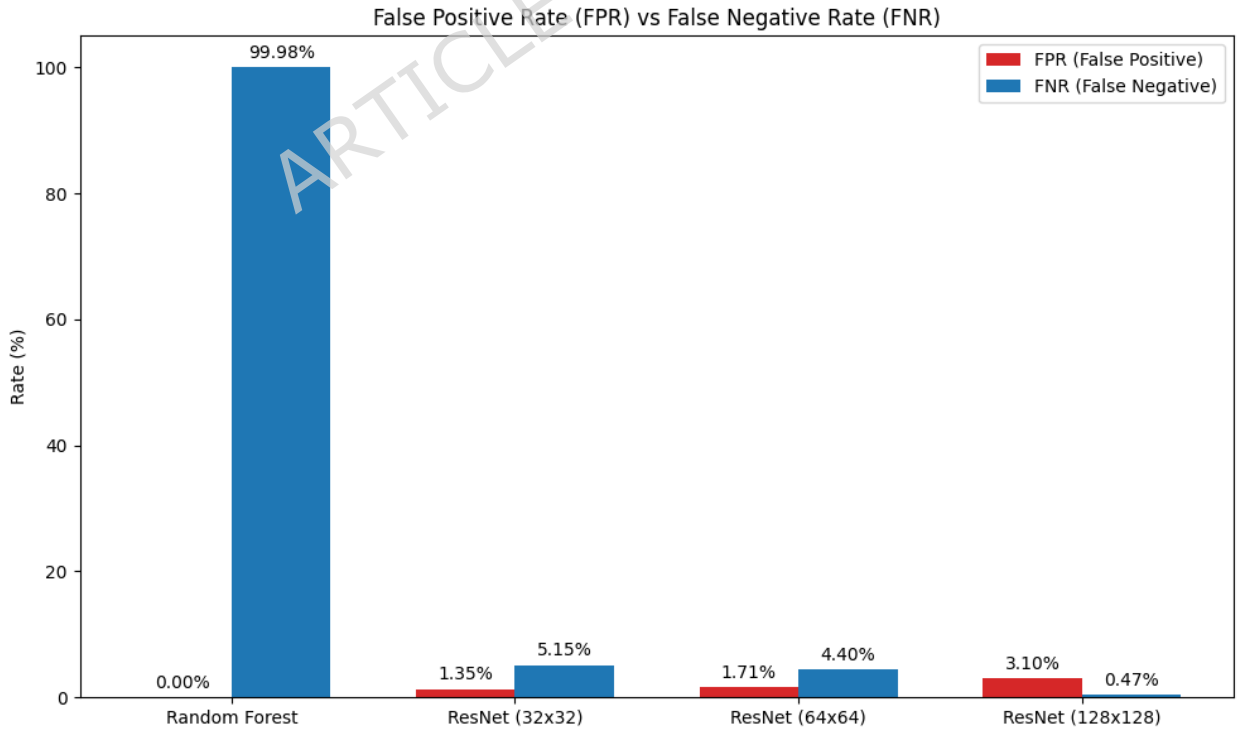

FIGURE 7. False Positive Rate (FPR) and False Negative Rate (FNR) on the Cross-Scenario Test

Figure 7 also reveals a critical trade-off inherent in detection systems. As image size increases, the FNR steadily decreases, meaning the model becomes more sensitive at detecting every possible threat (down to an impressive 0.47% for the 128x128 model). However, this comes at the direct cost of a rising FPR, which more than doubles from the 32x32 to the 128x128 configuration, leading to more false alarms for security analysts to investigate.

4.2. Discussion

The Role of Sparsity and Representation: A valid concern regarding our traffic-to-image conversion is the sparsity of the resulting data. With a fixed feature vector length of **19** and a target image size of 32×32 (1024 pixels), approximately 98% of the image pixels are zeros (black). However, in the context of Deep Learning, this sparsity does not hinder performance. CNNs are translation-invariant and excel at detecting local patterns regardless of their position in the frame. The critical contribution of the Hilbert curve is that it clusters the non-zero pixels (active features) into spatially coherent regions. This creates distinctive "glyphs" or local textures that represent specific traffic behaviors. The **ResNet-18** architecture, with its deep residual connections, effectively learns to identify these sparse but structured patterns, much like how it might identify edges in a minimal sketch.

Comparison with Threshold-Based Baselines: Our comparison with the Random Forest (RF) baseline serves as a proxy for the limitations of threshold-based statistical learning in unseen scenarios. While modern gradient boosting methods (like XGBoost) or 1D-CNNs might offer marginal improvements over RF on *known* data, they often suffer from the same fundamental flaw when facing concept drift: they rely on specific numerical thresholds (e.g., "if bytes > 500"). When a new botnet family (like Rbot) shifts these statistical distributions, threshold-based models fail catastrophically, as evidenced by the RF's 0.02% recall. In contrast, our Image-based approach creates a topological fingerprint. Even if the absolute byte counts shift, the *relative relationship* between features (the shape of the curve) often remains consistent, allowing the 2D-CNN to generalize where 1D statistical models fail.

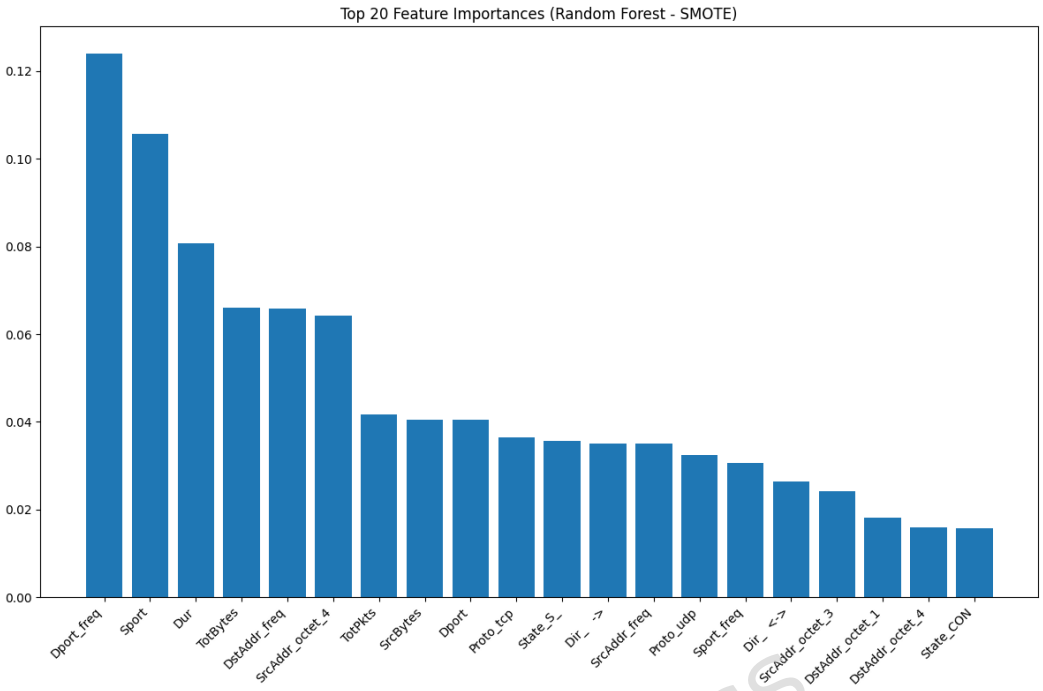

FIGURE 8. Top 20 Feature Importances for the Random Forest Model. The features were learned exclusively from the Scenario 8 (Murlo) training data, highlighting the model's reliance on specific statistical properties of that particular botnet.

Figure [8] illustrates this critical dependency by showing the top 20 features the model relied upon most heavily during its training on Scenario 8. The model's decisions were dominated by highly specific statistical artifacts of the Murlo botnet's traffic, such as destination port frequency (Dport_freq), flow duration (Dur), and total bytes (TotBytes). Essentially, the model did not learn a general concept of "maliciousness"; instead, it memorized the specific profile of the Murlo botnet.

5. Conclusion

In this study, we moved beyond conventional botnet detection methods by proposing a robust deep learning framework centered on advanced representation learning. By replacing simplistic encoding and padding techniques with more sophisticated methods like **octet splitting and the Hilbert space-filling curve**, we created a richer, more spatially coherent data representation. Crucially, by employing a challenging **cross-scenario validation** methodology, we shifted the focus from mere classification accuracy to the critical measure of model generalization—the ability to detect threats never seen before.

Our model demonstrated exceptional performance, achieving a peak accuracy of **98.34%** in detecting a completely unseen botnet family, a task where a traditional Random Forest classifier failed entirely due to severe overfitting. This result underscores the capability of our representation-first approach to function as a practical defense against **zero-day threats**, learning the fundamental patterns of malicious behavior rather than the superficial artifacts of a specific attack.

While this study provides a robust proof-of-concept, it also lays the foundation for future innovation. Because our method generalizes from metadata and flow characteristics rather than inspecting payloads, it is inherently well-suited for analyzing **encrypted traffic** and the heterogeneous data streams from **Internet of Things (IoT)** ecosystems. Future work should explore creating even richer, multi-channel image representations, where each channel (e.g., R, G, B) encodes a different aspect of the network flow (timing, packet size, protocol flags). Furthermore, the application of more advanced architectures like **Vision Transformers (ViT)**, which excel at capturing global relationships within images, could further enhance detection capabilities against even more sophisticated and evasive threats. This research offers more than just a new tool; it provides a new lens through which to view network data, demonstrating that by creatively reimagining the nature of the information we seek to protect, we can build a more secure and resilient digital future.

Acknowledgments

We extend our gratitude to the creators of the CTU-13 dataset at the *Stratosphere Laboratory* for making their valuable data publicly available to the research community.

Funding Declaration

The author received no financial support for the research, authorship, and/or publication of this article.

Data Availability

The dataset analyzed during the current study is the CTU-13 dataset, which is publicly available in the Stratosphere Laboratory repository, https://www.stratosphereips.org/datasets-ctu13.

Code Availability

The source code, pre-processing scripts, and model architectures used to generate the results in this study are available in the GitHub repository: https://github.com/occbuu/RepLearningDetectBotnetAttack.

References

1. Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1). https://doi.org/10.1002/ett.4150

2. Ahmed, A., & Tjortjis, C. (2022). Machine learning based IoT-BotNet attack detection using real-time heterogeneous data. *2022 International Conference on Electrical, Computer and Energy Technologies (ICECET)*.

3. Ahmed, U., Nazir, M., Sarwar, A., Ali, T., Aggoune, E.-H. M., Shahzad, T., & Khan, M. A. (2025). Signature-based intrusion detection using machine learning and deep learning approaches empowered with fuzzy clustering. *Scientific Reports,* 15(1), 1726. https://doi.org/10.1038/s41598-025-85866-7

4. Akash, N. S., Rouf, S., Jahan, S., Chowdhury, A., & Uddin, J. (2022). Botnet detection in IoT devices using Random Forest Classifier with Independent Component Analysis. *Journal of ICT,* 21(2), 201–232. https://doi.org/10.32890/jict2022.21.2.3

5. Ali, S., Ghazal, R., Qadeer, N., Saidani, O., Alhayan, F., Masood, A., Saleem, R., Khan, M. A., & Gupta, D. (2024). A novel approach of botnet detection using hybrid deep learning for enhancing security in IoT networks. *Alexandria Engineering Journal,* 103, 88–97. https://doi.org/10.1016/j.aej.2024.05.113

6. Aljehane, N. O., Mengash, H. A., Hassine, S. B. H., Alotaibi, F. A., Salama, A. S., & Abdelbagi, S. (2024). Optimizing intrusion detection using intelligent feature selection

with machine learning model. *Alexandria Engineering Journal,* 91, 39–49. https://doi.org/10.1016/j.aej.2024.01.073

7. Alkahtani, H., & Aldhyani, T. H. H. (2021). Botnet attack detection by using CNN-LSTM model for Internet of Things applications. *Security and Communication Networks,* 2021, 1–23. https://doi.org/10.1155/2021/3806459

8. Alnajim, A., Habib, S., Islam, M., Thwin, S., & Alotaibi, F. (2023). A comprehensive survey of cybersecurity threats, attacks, and effective countermeasures in Industrial Internet of Things. *Technologies*, 11(6), 161. https://doi.org/10.3390/technologies11060161

9. Alomari, D., Anis, F., Alabdullatif, M., & Aljamaan, H. (2023). A survey on botnets attack detection utilizing machine and deep learning models. *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering*, 493–498.

10. Alshamkhany, M., Alshamkhany, W., Mansour, M., Khan, M., Dhou, S., & Aloul, F. (2020). Botnet Attack Detection using Machine Learning. *2020 14th International Conference on Innovations in Information Technology (IIT).*

11. Al-Shurbaji, T., Anbar, M., Manickam, S., Hasbullah, I. H., Alfriehat, N., Alabsi, B. A., Alzighaibi, A. R., & Hashim, H. (2025). Deep learning-based intrusion detection system for detecting IoT botnet attacks: A review. *IEEE Access,* 13, 11792–11822. https://doi.org/10.1109/access.2025.3526711

12. Alwhbi, I. A., Zou, C. C., & Alharbi, R. N. (2024). Encrypted network traffic analysis and classification utilizing machine learning. *Sensors (Basel, Switzerland),* 24(11), 3509. https://doi.org/10.3390/s24113509

13. Aversano, L., Bernardi, M. L., Cimitile, M., & Pecori, R. (2021). A systematic review on Deep Learning approaches for IoT security. *Computer Science Review*, 40(100389), 100389. https://doi.org/10.1016/j.cosrev.2021.100389

14. Awad, L., Al-HajBaddar, S., & Sleit, A. (2023). Machine learning and deep learning for botnet detection techniques: A comparative review. *In Lecture Notes in Networks and Systems* (pp. 401–412). Springer Nature Switzerland.

15. Bakour, K., & Ünver, H. M. (2021). DeepVisDroid: android malware detection by hybridizing image-based features with deep learning techniques. *Neural Computing & Applications*, 33(18), 11499–11516. https://doi.org/10.1007/s00521-021-05816-y

16. Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, 18(2), 1153–1176. https://doi.org/10.1109/comst.2015.2494502

17. Chen, Z. (2021). Detection of IoT botnets using decision trees (Doctoral dissertation). ProQuest Dissertations and Theses. https://www.proquest.com/openview/0e58828dd5d3ff63e47cbc0618cd1baf/1?pq-origsite=gscholar&cbl=18750&diss=y

18. Dasgupta, D., Akhtar, Z., & Sen, S. (2022). Machine learning in cybersecurity: a comprehensive survey. *The Journal of Defense Modeling and Simulation Applications Methodology Technology*, 19(1), 57–106. https://doi.org/10.1177/1548512920951275

19. Emirmahmutoğlu, E., & Atay, Y. (2025). A feature selection-driven machine learning framework for anomaly-based intrusion detection systems. *Peer-to-Peer Networking and Applications*, 18(3). https://doi.org/10.1007/s12083-025-01947-4

20. Fang, W. (2023). Real time botnet detection system based on machine learning algorithms. In L. Shen (Ed.), *2022 2nd Conference on High Performance Computing and Communication Engineering (HPCCE 2022)* (p. 44). SPIE.

21. Garcia, S., Grill, M., Stiborek, J., & Zunino, A. (2014). An empirical comparison of botnet detection methods. *Computers & Security*, 45, 100-123.

22. Gueriani, A., Kheddar, H., & Mazari, A. C. (2024). Enhancing IoT security with CNN and LSTM-based intrusion detection systems. *2024 6th International Conference on Pattern Analysis and Intelligent Systems (PAIS).*

23. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 770–778. https://doi.org/10.1109/CVPR.2016.90

24. Ho, C. M. K., Yow, K.-C., Zhu, Z., & Aravamuthan, S. (2022). Network intrusion detection via flow-to-image conversion and vision transformer classification. *IEEE Access,* 10, 97780–97793. https://doi.org/10.1109/access.2022.3200034

25. Hoang, X. D., & Vu, X. H. (2022). An improved model for detecting DGA botnets using random forest algorithm. *Information Security Journal A Global Perspective*, 31(4), 441–450. https://doi.org/10.1080/19393555.2021.1934198

26. Janabi, A. H., Kanakis, T., & Johnson, M. (2024). Survey: Intrusion detection system in software-defined networking. *IEEE Access*, 12, 164097–164120. https://doi.org/10.1109/access.2024.3493384

27. Kasongo, S. M., & Sun, Y. (2019). A deep learning method with filter based feature engineering for wireless intrusion detection system. *IEEE Access,* 7, 38597–38607. https://doi.org/10.1109/access.2019.2905633

28. Kim, T., & Pak, W. (2023). Deep learning-based network intrusion detection using multiple image transformers. *Applied Sciences (Basel, Switzerland),* 13(5), 2754. https://doi.org/10.3390/app13052754

29. Kundu, P. P., Truong-Huu, T., Chen, L., Zhou, L., & Teo, S. G. (2024). Detection and classification of botnet traffic using deep learning with model explanation. *IEEE Transactions on Dependable and Secure Computing,* 1–15. https://doi.org/10.1109/tdsc.2022.3183361

30. Mijwil, M. M., Salem, I. E., & Ismaeel, M. M. (2023). The significance of machine learning and deep learning techniques in cybersecurity: A comprehensive review. *Iraqi Journal for Computer Science and Mathematics,* 87–101. https://doi.org/10.52866/ijcsm.2023.01.01.008

31. Mohale, V. Z., & Obagbuwa, I. C. (2025). Evaluating machine learning-based intrusion detection systems with explainable AI: enhancing transparency and interpretability. *Frontiers in Computer Science,* 7(1520741). https://doi.org/10.3389/fcomp.2025.1520741

32. Mothukuri, V., Khare, P., Parizi, R. M., Pouriyeh, S., Dehghantanha, A., & Srivastava, G. (2022). Federated-learning-based anomaly detection for IoT security attacks. *IEEE Internet of Things Journal,* 9(4), 2545–2554. https://doi.org/10.1109/jiot.2021.3077803

33. Nketia, I. K., Yaokumah, W., & Appati, J. K. (2024). A comprehensive review of internet-of-things (IoT) botnet detection techniques. *In Smart and Agile Cybersecurity for IoT and IIoT Environments* (pp. 50–81). IGI Global.

34. Nguyen, T. D., Marchal, S., Miettinen, M., Fereidooni, H., Asokan, N., & Sadeghi, A.-R. (2019). DÏoT: A federated self-learning anomaly detection system for IoT. *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS).*

35. Pokhrel, S., Abbas, R., & Aryal, B. (2021). IoT Security: Botnet detection in IoT using Machine learning. *arXiv:2104.02231.* https://doi.org/10.48550/ARXIV.2104.02231

36. Qazi, E. U. H., Almorjan, A., & Zia, T. (2022). A one-dimensional convolutional neural network (1D-CNN) based deep learning system for network intrusion detection. *Applied Sciences (Basel, Switzerland),* 12(16), 7986. https://doi.org/10.3390/app12167986

37. Raghava, N. S., & Sahgal, D. (2012). Botnet: A deadly threat to cyber security. *International Journal of Computer Applications*, 44(2).

38. Ren, F., Jiang, Z., Wang, X., & Liu, J. (2020). A DGA domain names detection modeling method based on integrating an attention mechanism and deep neural network. *Cybersecurity*, 3(1). https://doi.org/10.1186/s42400-020-00046-6

39. Rodríguez, E., Valls, P., Otero, B., Costa, J. J., Verdú, J., Pajuelo, M. A., & Canal, R. (2022). Transfer-learning-based intrusion detection framework in IoT networks. *Sensors (Basel, Switzerland),* 22(15), 5621. https://doi.org/10.3390/s22155621

40. Saeed, M., Saeed, R., Abdelhaq, M., Alsaqour, R., Hasan, M., & Mokhtar, R. (2023). Anomaly detection in 6G networks using machine learning methods. Electronics, 12(15), 3300. https://doi.org/10.3390/electronics12153300

41. Shen, M., Ye, K., Liu, X., Zhu, L., Kang, J., Yu, S., Li, Q., & Xu, K. (2023). Machine learning-powered encrypted network traffic analysis: A comprehensive survey. *IEEE Communications Surveys & Tutorials,* 25(1), 791–824. https://doi.org/10.1109/comst.2022.3208196

42. Soofi, A. A., Tahir, M., & Raza, N. (2024). Securing the Internet of Things: A comprehensive review of security challenges and artificial intelligence solutions. *Foundation University Journal of Engineering and Applied Sciences*, 4(2), 1–20. https://doi.org/10.33897/fujeas.v4i2.779

43. Suthar, F., Patel, N., & Khanna, S. V. O. (2022). A signature-based botnet (emotet) detection mechanism. *International Journal of Engineering Trends and Technology*, 70(5), 185–193. https://doi.org/10.14445/22315381/ijett-v70i5p220

44. Vinayakumar, R., Alazab, M., Soman, K. P., Poongodi, P., & Al-Turjman, F. (2021). *Deep learning for cyber security applications: A comprehensive survey.* https://doi.org/10.36227/techrxiv.16748161.v1

45. Wang, W., Jian, S., Tan, Y., Wu, Q., & Huang, C. (2022). Representation learning-based network intrusion detection system by capturing explicit and implicit feature interactions. *Computers & Security*, 112(102537), 102537. https://doi.org/10.1016/j.cose.2021.102537

46. Xing, Y., Shu, H., Zhao, H., Li, D., & Guo, L. (2021). Survey on botnet detection techniques: Classification, methods, and evaluation. *Mathematical Problems in Engineering,* 2021, 1–24. https://doi.org/10.1155/2021/6640499