IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 23 April 2024, accepted 14 May 2024, date of publication 22 May 2024, date of current version 31 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3404270

RESEARCH ARTICLE

Network Traffic Visualization Coupled With Convolutional Neural Networks for Enhanced IoT Botnet Detection

DAVID ARNOLD, (Member, IEEE), MIKHAIL GROMOV, (Member, IEEE), AND JAFAR SANIIE, (Life Fellow, IEEE)
Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL 60616, USA
Corresponding author: Jafar Saniie (saniie@iit.edu)

ABSTRACT Systemic vulnerabilities in the Internet of Things (IoT) pose a challenge for establishing robust cybersecurity strategies. These challenges leave IoT devices susceptible to infection, often falling victim to far-reaching Botnets. To counter these risks, Intrusion Detection Systems (IDS) are designed to detect attacks within the network, mitigating the dangers presented by architecturally vulnerable IoT devices. However, IDS solutions are designed to operate at the center of the network, requiring network traffic to be forwarded inwards and consequently hampers reaction times while straining network resources. This paper introduces an IoT Botnet detection pipeline composed of a novel network traffic visualization methodology and a Convolutional Neural Network (CNN). The pipeline operates on an embedded system at the edge of the network, transforming network traffic into a visual format for subsequent cyberattack classification by the CNN. By leveraging the advantages of CNNs in efficiently classifying images, the pipeline achieves high accuracy in detecting Botnet attacks while maintaining an efficient design. During testing, we applied the pipeline to the N-BaIoT and IoT-23 datasets and observed high cyberattack detection rates of 100% and 99.78%, respectively. Furthermore, we observed a 2.4 times greater throughput (packets/second) and a 21.4% reduction in model size compared to a Deep Neural Network of similar accuracy.

INDEX TERMS Botnets, cybersecurity, convolutional neural network, intrusion detection systems.

I. INTRODUCTION

The Internet of Things (IoT) continues to fuel innovation and growth across the medical, industrial, and energy sectors [1], [2]. According to IoT Analytics' "State of IoT-Spring 2023" report, the number of IoT devices grew by 18% in 2022 to 14.3 billion active devices, with the growth expected to dip to 16% in 2023 [3]. The firm also predicts that the enterprise IoT market will grow from $201 billion in 2022 to $483 billion by 2027. In their Annual Internal Report (for 2018-2023), Cisco predicts that by the end of 2023, 50% of all global networked devices will be IoT [4]. Despite this strong growth, integrating IoT devices introduces challenges for applying strong cybersecurity controls within the network.

Alongside their growth, IoT systems have experienced severe difficulties in fending off cyberattacks [5], [6], [7], [8], [9]. As part of their efforts to enhance Internet security, the Open Worldwide Application Security Project (OWASP) publishes a ranking of common cyberattacks against IoT in their OWASP Top 10 Internet of Things. Within their 2018 rankings, vulnerabilities such as 1) Weak, Guessable, or Hardcoded Passwords, 2) Insecure Network Services, 3) Insecure Ecosystem Interfaces, and 4) Lack of Secure Update Mechanisms were identified as areas of concern [10]. The simple nature of these vulnerabilities highlights the current deficiencies in IoT cybersecurity. Factors that contribute to the insecurity of IoT include the increasing heterogeneity of network infrastructure and a lack of secure coding principles [11], [12]. In many applications, numerous vendors and devices are required to maximize the use of distributed sensors and cyber-physical systems. This influx of unique

The associate editor coordinating the review of this manuscript and approving it for publication was Guangjie Han.

devices complicates the cohesive application of cybersecurity policies throughout the network. The ongoing lack of cybersecurity standards further reduces adherence to secure coding practices during development. Limited computation and authentication capabilities also contribute, weakening resistance to eavesdropping, impersonation, and Denial of Service (DoS) attacks [13], [14], [15], [16]. Altogether, these weaknesses create the perfect conditions for compromising vast numbers of devices.

Due to the simplicity of attacks against IoT devices, it wasn't long before hackers began gathering large numbers of compromised devices into larger networks known as botnets. Infection of a device starts when a scanner identifies a potentially vulnerable device and reports it to a central database [17], [18]. Depending on the sophistication of the botnet, the central server will either attempt to brute force the SSH login or exploit a known vulnerability on the device. After securing access, the botnet will prevent other hackers from gaining access by turning off remote access features and will contact the command and control (C2) server to download architecture-specific malware [19]. After this occurs, the device may attempt to scan for additional devices for infection, contribute towards a Distributed Denial of Service (DDoS) attack, mine for cryptocurrency, or maybe shut down. Botnet developers constantly update their botnets for enhanced evasion or infection. The most well-known botnet was the Mirai botnet, which achieved a peak of 600,00 infections in 2016 and improved the Bashlite botnet [17]. Advancements in the Mirai botnet included TCP SYN scans based on pseudorandom IPv4 addresses for more efficient victim identification. In 2023, the IZ1H9 botnet, a Mirai-based botnet, was observed incorporating thirteen new payloads to account for new vulnerabilities identified in IoT devices [20]. Additionally, some botnet architectures are shifting away from centralized C2 infrastructures towards peer-to-peer (P2P) architectures to complicate eradicating the botnet's control system [21]. Due to the dangers posed by these botnets, Intrusion Detection Systems (IDS) must identify botnet activity to prevent infection and isolate infected devices.

Machine learning models are effective tools in identifying cyberattacks and are frequently found in IDS roles [22], [23], [24], [25], [26]. Compared to rule-based IDSs, ML models are trained to identify abnormal behavior and signatures of common attack types. Current solutions operate within centralized network architectures, drawing from deep resource pools. For IoT devices, traffic must be forwarded internally for processing, straining network resources and harming response time. To improve response times, an IDS can be introduced at the network's edge; however, to remain cost-effective, the IDS must be hosted on an embedded system. This is challenging as many modern ML models have high parameter counts, necessitating large storage volumes, which are infeasible on low-cost embedded systems.

To address the limitations of deploying machine learning within an IoT environment, we introduce a lightweight IoT Botnet Detection Pipeline. This pipeline is composed of 2 stages: 1) a novel network traffic visualization methodology and 2) a Convolutional Neural Network for network traffic classification. By leveraging the advantages of the CNN in classifying images, we achieve a 2.4 times greater throughput compared to a Deep Neural Network and a reduction in the number of trainable parameters by 21.4%. When coupled with a Jetson Nano, the IoT Botnet Detection Pipeline keeps pace with the throughput demands of IoT environments at a low cost.

This paper contributes to the IoT cybersecurity literature in the following manner:

1. Proposes a novel network traffic visualization methodology for transforming network traffic into a visual format.
2. Introduces a Convolutional Neural Network model for classifying visualized network traffic.
3. Compares the proposed IoT Botnet Detection Pipeline against a Deep Neural Network (DNN) model and Autoencoder (AE) model using two IoT Datasets: N-BaIoT [27] and IoT-23 [28]. The following metrics are used:
   a. Botnet Detection Accuracy
   b. Botnet Classification Accuracy
   c. Throughput (packets/second)
   d. Model Parameter Count

This paper is organized as follows. In Section II, we will discuss related works and explore alternative solutions for detecting botnet activity within an IoT network. Next, Section III will introduce the visualization methodology and present examples of visualized network traffic. Section IV will examine our machine learning models, including a Convolutional Neural Network, Deep Neural Network, and Autoencoder models. Botnet detection and classification accuracy are presented and discussed in Section V. Finally, we will summarize our work in Section VI.

II. RELATED WORKS

There are many methods for detecting and preventing Botnet infections and attacks within IoT ecosystems. For instance, a hardening script may prevent the Botnet from executing its payload [29], or an automated scanning tool may search for Botnet fingerprints [30]. While these tactics are effective, they rely on a database of signatures and may not adapt to novel threats. In the hopes of adapting to novel attack flows, research has shifted towards applying anomaly detection for identifying Botnet activity. For instance, an analysis of Blockchain transactions among IoT devices may reveal deviations from normal activity [31], [32], [33]. However, most of the research has been centered around applying deep learning models for implementing intrusion detection capabilities. Advances in Deep Learning and the general availability of high-powered computing resources have made these solutions viable within traditional networks. Ongoing research into these solutions spans from i) Deep Neural Networks (DNN), ii) Convolutional Neural Networks (CNN), iii) Autoencoders (AE) and Generative Adversarial Models (GAN), and iv) Long-Short Term Memory (LSTM) models.

Beginning with the least complex model, Deep Neural Networks are a family of machine learning models where weights and biases are trained based on the input and classification. Huma et al. [34] explored the potential application of a Deep Neural Network composed of 3 Recurrent Neural Networks (RNN) and 3 Multilayer Perceptron Layers (MLP) within the context of an Industrial Internet of Things (IIoT) setting. Their research found that the model successfully detected cyberattacks within the DS2OS (98% accuracy) and UNSW-NB15 (99% accuracy) datasets.

Next, Convolutional Neural Networks are another popular class of Deep Learning models. Typically applied against a matrix or image input, these models apply convolutions with kernels and biases during classification. The most substantial benefit of these architectures is that they require fewer parameters when compared to a Deep Neural Network of similar complexity. Research regarding a CNN approach was conducted by Hussain et al. [35], in which they developed an ensemble approach for detecting Denial of Service (DoS) and scanning activity and achieved an overall accuracy of 98.89%. Additional research into CNN for botnet detection includes [36], [37]. In [36], Ullah et al. explored the BoT-IoT, MQTT-IoT-IDS2020, and IoT-23 datasets using 1D, 2D, and 3D Convolutional Networks. In [37], Hairab et al. applied a CNN model against the Bot-IoT dataset, achieving 91% accuracy.

Autoencoders and Generative Adversarial Networks are primarily used for expanding datasets with data of a similar form; however, when a discriminator is introduced, they can be used in classification and detection applications. Abdalgawad et al. [38] explored a potential solution using an Adversarial Autoencoder (AAE) and a Bidirectional General Adversarial Network (BiGAN) against the IoT-23 dataset, which we will be using as well. During their research, they found that they were superior to basic Machine Learning models (Random Forest, Support Vector Machine....) and achieved an F1-Score of 0.99 for both the AAE and BiGAN models. In addition to [38], the developers of the N-BaIoT dataset, Meidan et al. [27], used an Autoencoder as a classifier in their research. In their implementation, they used four hidden layers in the encoder and decoder, which decreased the input size to 75%, 50%, 33%, and 25% after each layer. Further, they trained models for each device within the environment and conducted cyberattack detection, achieving near-perfect accuracy. In contrast, we will develop a multi-class classification Autoencoder model for generalized cyberattack detection across a diverse range of devices. We discuss the differences between our Autoencoder implementation and the N-BaIoT implementation in Section IV.

Long-Short Term Memory (LSTM) models are another popular model for detecting botnet attacks. These models are popular as they inject some knowledge regarding previously recorded traffic. This is beneficial as many cyberattacks have multiple stages and can be challenging to detect when examining individual points in time. Alkahtani and Aldhyani [39] explored a classification model that combined a CNN and an LSTM model. Throughout their research, they ran their classifier over the N-BaIoT dataset and achieved an accuracy of 100%. Other papers that applied an LSTM included [40], [41]. In [40], Saharkhizan et al. designed an ensemble method, combining multiple LSTM models for cyberattack detection in Modbus network traffic. Through their methodology, they achieved an accuracy of 99.62%.

Regarding the application of visualization methodologies, data visualization as a pre-processing tool is not a new concept in cybersecurity. The authors in [42] extracted the binary image of infected IoT devices and converted them to an image format prior to analysis. After analysis, the team achieved 94% accuracy when working with DDoS attacks and 81.8% accuracy for main malware families. Next, the authors in [43] converted the payload of encrypted traffic to identify malware. Using the ISCX VPN-NonVPN dataset, the team achieved an F1 Score of 97.73% in detecting malware. Finally, the work most similar to ours was [44]. In this work, the authors converted the raw network traffic before classification using the ResNet50 CNN to achieve a 94.50% accuracy in their self-generated dataset. Our work differs as we extract features from the packet metadata before analysis. Further, we focused on developing a lightweight classifier for applications within the network's edge.

Regarding our previous research, we explored botnet detection and classification within the N-BaIoT and IoT-23 datasets. During our research on the N-BaIoT dataset, we explored the heterogeneity tolerance of Deep Neural Networks when classifying Miria and Gafgyt traffic within a diverse environment of IoT devices [45]. Additionally, we explored the potential application of an Autoencoder [46] and a Convolutional Neural Network [47] for conducting binary classification of botnet attacks in the N-BaIoT dataset. Lastly, we explored the application of a CNN on the IoT-23 dataset [48].

Our work expands on existing solutions by exploring the effectiveness of a visualization methodology in improving botnet detection at the network's edge. In conjunction with a Convolutional Neural Network, we achieve high accuracy in detecting botnet attacks. Simultaneously, our pipeline has high throughput and a low parameter count, ideal for embedded systems. We compare our results with a deep neural network and autoencoder of similar accuracy. Our Deep Neural Network approach will not utilize the visualization methodology and will serve as a baseline comparison. On the other hand, our Autoencoder was designed for multi-class classification and presented an alternative approach to the model proposed by Meidan et al. in the N-BaIoT paper [27].

III. VISUALIZATION METHODOLOGY

The proposed network traffic visualization methodology converts network traffic into a visual format for classification by a Convolutional Neural Network. To achieve this, the methodology extracts metadata and packet statistics from each incoming packet. The following metadata is extracted: 1) Source MAC Address, 2) Source IP Address, 3) Destination IP Address, and 4) Destination Port Number. The

extracted packet statistics include 1) The average size of outbound packets, 2) The average size of both inbound and outbound packets, 3) The number of packets from the Source, and 4) The average time (jitter) between arrivals. This information is then used to generate 12 features for our image generation. The reasoning behind our features and how to calculate them will be discussed in Section III-A. After generating our 12 features, the process by which they are organized into a visual format will be presented in Section III-B.

A. DATASETS

During our research, we evaluated our pipeline against two publicly available datasets, N-BaIoT and IoT-23. First, the N-BaIoT dataset was collected by Meidan et al. in 2018, examining the Mirai and Gafgyt (also known as BASHLITE) botnets. We selected this dataset as it contained a heterogeneous network of 9 consumer IoT devices during both botnets' propagation and attack stages. Additionally, the authors considered sub-classifications of Mirai and Gafgyt activity, providing 5 Mirai and 5 Gafgyt sub-classifications. The inclusion of sub-classifications further refines our identification of the botnets and indicates the infection stage from which the IoT device is suffering.

Compared to alternative datasets, N-BaIoT consists of network traffic metadata statistics rather than raw network traffic. To transfer machine learning models based on the methodology introduced by Meidan, we need to complete feature extraction on raw network traffic data. This is accomplished by collecting each packet's MAC address, Source IP, Destination IP, and port number. Meidan et al. collected these statistics over 5 time windows: 100 milliseconds, 500 milliseconds, 1.5 seconds, 10 seconds, and 1 minute. Each time window accounted for 23 features, resulting in an overall feature set of 115 features. However, we focused on only the smallest time decay for our application, reducing our features to 23. To optimize image generation, we further decreased the feature set to 12 features, resulting in 2×2 blocks with 3 channels for an RGB image. To select the 12 remaining features, we generated images for Benign and Mirai traffic under different combinations of features. Our final features were selected under 2 criteria: 1) the features created images with significant variation between the Mirai and Benign images, and 2) the features were not similar to already selected features. Table 1 contains the final 12 features used in our visualization methodology, with labels referring to the label applied in the original N-BaIoT dataset.

After evaluating our visualization methodology against the N-BaIoT dataset, we explored the IoT-23 dataset to validate our methodology. Published in 2020 by Garcia et al., the IoT-23 dataset examines 3 IoT devices and generated 23 operating scenarios, comprised of 3 benign and 20 malicious scenarios. Within the malicious scenarios, the authors studied 11 classes of cyberattacks, including the Mirai Botnet, the Torii Botnet, the Okiru Botnet, and a Trojan Malware sample. A complete list of classes for the IoT-23 dataset is presented in Table 2. Unlike the N-BaIoT dataset, IoT-23 provides raw network traffic in a .pcap file and a labeled Zeek flow, which classifies the network traffic. Since our methodology was initially developed around the N-BaIoT dataset, we extracted the metadata features from the raw traffic before applying the visualization methodology. As stated, the MAC Address, Source IP, Destination IP, and port number were collected, and our 12 features were generated for each packet.

TABLE 1. Features extracted for the visualization methodology.

N-BaIoT Index [27]	Feature	Description
0	MI_dir_L5_weight	Packet count based on source MAC-IP
1	MI_dir_L5_mean	The average size of packets based on source MAC-IP
2	MI_dir_L5_variance	Variance of packet size based on source MAC-IP
15	H_L5_weight	Packet count based on source IP
16	H_L5_mean	The average size of outbound packets
17	H_l5_variance	Variance of outbound packets
31	HH_L5_mean	The average size of outbound packets based on destination IP
33	HH_L5_magnitude	Packet size of inbound and outbound packets to a destination IP
34	HH_L5_radius	Root Squared sum of inbound and outbound packets to a destination IP
66	HH_jit_L5_mean	Average jitter (time between arrivals) for packets based on destination IP
81	HpHp_L5_mean	Average packet size based on IP and port
82	HpHp_L5_std	The standard deviation of packet size based on IP and port

TABLE 2. N-BaIoT and IoT-23 classes.

	N-BaIoT Classes	IoT-23 Classes
0	Benign	Benign
1	Gafgyt - Combo	Mirai
2	Gafgyt – Junk	Torii
3	Gafgyt – Scan	Trojan
4	Gafgyt – TCP	Gafgyt
5	Gafgyt – UDP	Kenjiro
6	Mirai – ACK	Okiru
7	Mirai – Scan	Hakai
8	Mirai – SYN	IRCBot
9	Mirai – UDP	Hajime
10	Mirai – UDP plain	Muhstik
11		HideAndSeek

Both datasets exhibited an imbalance between Benign and Botnet activity. We took steps to balance these datasets and did not conduct testing to determine how these imbalances would impact our results; however, imbalances traditionally result in challenges in properly detecting the underrepresented classes. Within the N-BaIoT dataset, the Mirai Botnet represented the largest share (52.65%), the Gafgyt Botnet represented the second largest share (39.41%), and the Benign

data represented the smallest share (7.94%). For binary classification, we introduced a dropout layer to retain 20% of Botnet activity, and for multi-class classification, we took a random sample of 90,000 packets for each class (10,000 for each device). Regarding the IoT-23 dataset, the authors varied their evaluation periods with packet captures varying from 1 hour to 112 hours and containing anywhere from 23,000 to 271,000,000 packets. To balance the dataset, we took a 100,000-packet sample from each observed class.

The diverse range of devices and botnets presented in the N-BaIoT and IoT-23 make them an ideal pair for understanding the proposed pipeline's generalizability to future IoT environments. The N-BaIoT dataset offers a diverse range of devices, examining 9 consumer IoT devices, whereas the IoT-23 dataset explores a diverse range of botnets, studying 11 classes of botnet infection. While evaluation will be useful for detecting the presence of a botnet, the datasets lacked granularity to determine the type of activity undertaken by the botnet. For instance, the botnet may have attempted to identify vulnerable devices, exploit vulnerabilities, or establish command-and-control functionality. With this information, we may better understand the health of our network and fine-tune our response to the breach.

B. IMAGE GENERATION ALGORITHM

After calculating the 12 features according to Table 1, our visualization methodology converts each packet into a 2×2 pixel area. 12 features are grouped into 4 sets of 3 and assigned to the RGB channels of each pixel. This process is presented in Figure 1. While any number of features can be used, using an arrangement that does not fully populate the perfect square of each sub-image will require zero padding to optimize image generation. Each packet represents a sub-image, with consecutive packets joined together to form a complete image. As a result, each image represents a period between the first and last packet. To optimize performance, sub-images are organized into a perfect square. The process of assembling the completed image from sub-images is presented in Figure 2. Additionally, Algorithm 1 provides an implementation of image generation.

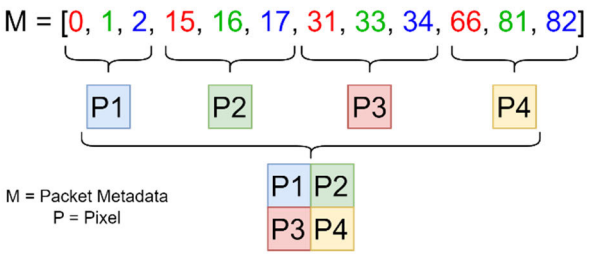


FIGURE 1. Sub-image creation. Metadata for each packet, composed of 12 features, is grouped into 4 groups of 3 for a 2 × 2 RGB sub-image.

For our analysis, 100 packets were used, forming a 20×20 RGB image. When considering the N-BaIoT dataset, we considered a time window of 100 milliseconds between packets. so that after every 100 milliseconds packets have half the impact on the moving average. While the original dataset contained 115 features, our previous research showed that a sub-sample of 12 features achieved the best results [32]. This provided an optimal throughput while maintaining comparable accuracy. A sample feature set from the N-BaIoT dataset is presented in Figure 3, containing samples collected from the Danmini Doorbell. Before applying our visualization methodology to the IoT-23 dataset, we needed to complete feature extraction. After doing so, we received the sample images presented in Figure 4.

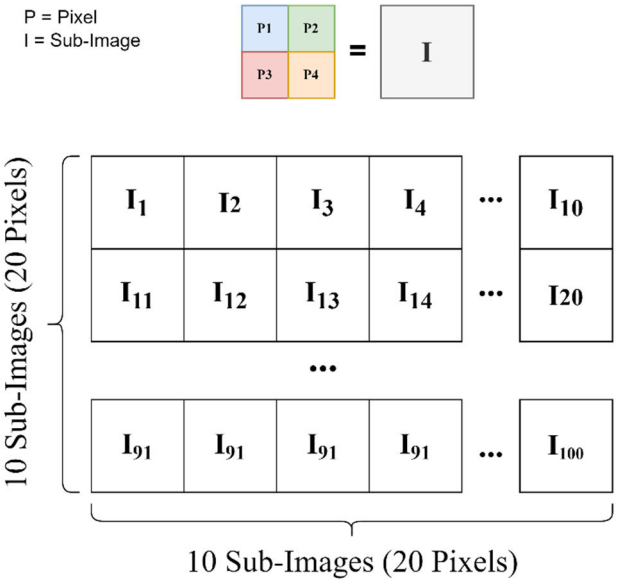


FIGURE 2. Network traffic visualization technique. After conversion into a sub-image, 100 packets are combined into a full image.

Algorithm 1 Visualization Methodology

INPUT:
p: Array of packets
n: Number of packets
OUTPUT:
image: image

```
Calculate a = √n
for i < n do
    for j < 3 do
        image[2∗(i//a)][2∗(i%a)][j] = p[i][j]
        image[2∗(i//a)][2∗(i%a)+1][j] = p[i][j+3]
        image[2∗(i//a)+1][2∗(i%a)][j] = p[i][j+6]
        image[2∗(i//a)+1][2∗(i%a)+1][j] = p[i][j+9]
    end for
end for
```

IV. MACHINE LEARNING MODELS

After developing our visualization methodology and transforming the N-BaIoT and IoT-23 datasets, we designed machine learning models for classifying the resulting images. Three classes of machine learning models were used for this evaluation: a Deep Neural Network (DNN), a Convolutional

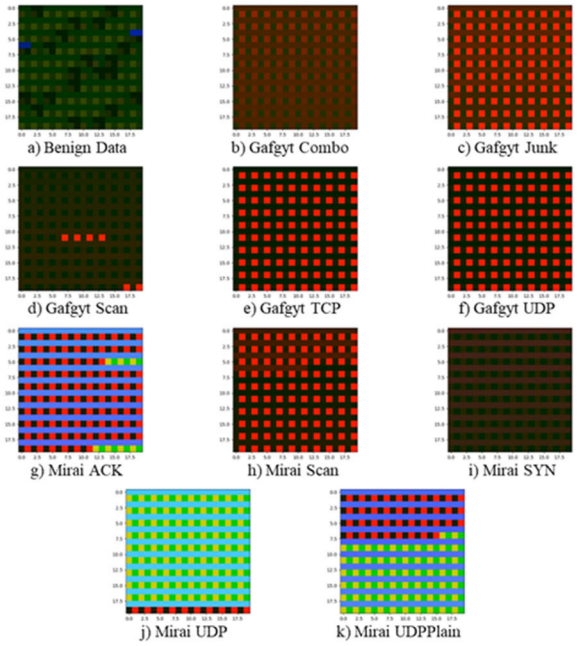


FIGURE 3. Sample traffic images generated from the N-BaIoT dataset.

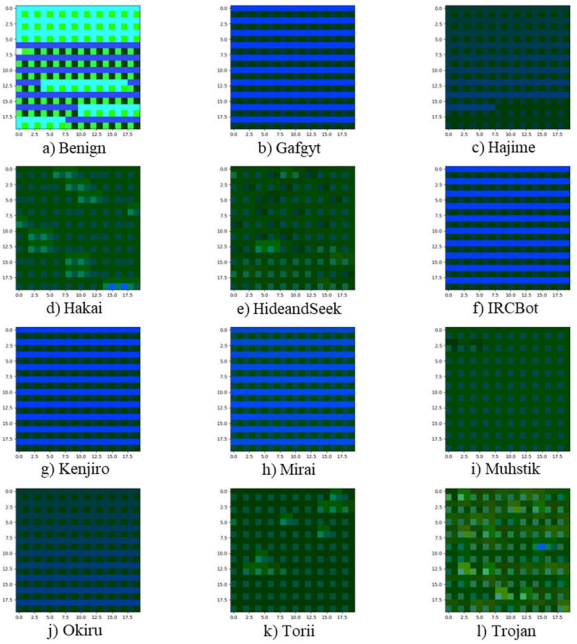


FIGURE 4. Sample traffic images generated from the IoT-23 dataset.

Neural Network (CNN), and an Autoencoder (AE). First, the Deep Neural Network was selected for its simplistic design and to serve as a baseline for evaluating the effectiveness of our methodology.

During training, the feature sets will not be transformed and will be supplied directly as inputs to our DNN models. Next, the Convolutional Neural Network was selected as it is popular for image analysis and classification within machine learning. Compared to DNNs of similar accuracy and complexity, CNNs have been found to have significantly reduced parameter counts. Finally, the Autoencoder was chosen as the initial N-BaIoT researchers applied an Autoencoder to evaluate their feature extraction methods. All three models were implemented in Python and utilized the TensorFlow libraries.

A. DEEP NEURAL NETWORK

Serving as our baseline evaluation model, Deep Neural Networks are simple machine learning models composed of layers of fully connected weights and biases, otherwise known as multilayer perceptron layers (MLP). Our model was based on our previous research regarding heterogeneity tolerance in Deep Neural Networks and was optimized for detecting cyberattacks within a diverse range of IoT devices [30]. Unlike our proposed methodology, the DNN uses the feature set from the N-BaIoT dataset, which is composed of 115 features.

Deep Neural Networks are composed of interconnected multilayer perceptron layers (MLP), in which the output of each layer is determined by multiplying and adding a set of weights and biases, respectively, followed by an activation function. When designing our model, we tuned the number of layers, the number of outputs for each layer, the activation function for each layer, and the regularization applied to each layer. To determine the best model, we optimized for accuracy and a low parameter count (the number of weights and biases). The final design for our DNN was composed of 4 layers:

1. Layer 1 – Input: 115, Output: 115, Activation: tanh, Regularization: L1 Regularization
2. Layer 2 – Input: 115, Output: 64, Activation: tanh, Regularization: None
3. Layer 3 – Input: 64, Output: 32, Activation: tanh, Regularization: None
4. Layer 4 – Input: 32, Output: 12, Activation: SoftMax, Regularization: None

The number of outputs for our final layer was based on the number of classes for each application: 1) Botnet Detection – 2 outputs, 2) Botnet Classification of the N-BaIoT dataset – 11 outputs, and 3) Botnet Classification of the IoT-23 dataset – 12 outputs. A visual representation of our DNN model is presented in Figure 5.

B. CONVOLUTIONAL NEURAL NETWORK

Our next model was a Convolutional Neural Network, a popular machine learning algorithm class for analyzing and classifying images. The models differ from deep neural networks in that they train a set of filters using the convolution function. Compared to the DNN, the convolution function greatly reduces the number of parameters that must be trained to achieve similar results.

The CNN is composed of a series of Convolutional Layers, which apply the Convolutional function with trained kernels against the input. IN between each Convolutional

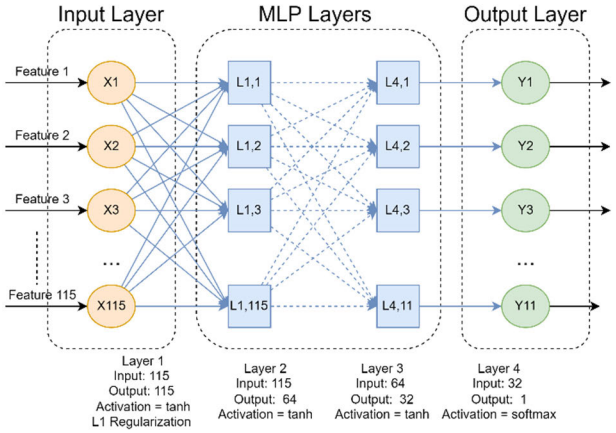


FIGURE 5. The deep neural network model for baseline comparison of our botnet intrusion detection system.

Layer, Max Pooling is used to reduce dimensionality and highlight important features in our images. For classification, the image is flattened and passed through a set of multilayer perceptron layers. During testing, we adjusted the number of Convolutional, Max Pooling, and MLP layers along with the Convolutional Kernel Size, the Convolutional Kernel Stride, the Convolutional Activation Function, the Max Pooling Kernel Size, the Output size of each MLP layer, and the MLP Activation Functions. After testing, we arrived at the following model using an input 20 × 20×3 image:

1. Convolutional Layer 1 – Kernel: 2 × 2×12, Stride: 2, Activation: ReLU
2. Max Pooling Layer 1 – Kernel: 2 × 2
3. Convolutional Layer 2 – Kernel: 2 × 2×16, Stride: 1, Activation: ReLu
4. Max Pooling Layer 2 – Kernel: 2 × 2
5. Flatten Layer
6. MLP Layer 1 – Input:64, Output: 128, Activation: Sigmoid
7. MLP Layer 2 – Input: 128, Output: 64, Activation: Sigmoid
8. MLP Layer 3 – Input: 64, Output: 12, Activation: Sigmoid

The number of outputs for our final layer was based on the number of classes for each application: 1) Botnet Detection – 2 outputs, 2) Botnet Classification of the N-BaIoT dataset – 11 outputs, and 3) Botnet Classification of the IoT-23 dataset – 12 outputs. A visual representation of our CNN model is presented in Figure 6.

C. AUTOENCODER

Lastly, we explored the application of an Autoencoder model for our cyberattack classification. We included this class of models as the N-BaIoT team used them to evaluate their feature extraction methodology when working with their dataset. As such, this allows us to compare the effectiveness of our methodology against existing research. Autoencoders attempt to reconstruct the input features according to a desired class. For instance, we may attempt to reconstruct a Benign image from the input data. For classification applications, we determine whether the input image is of the desired class by taking the error between the input and output images. If the error is low, we can conclude that the image is of the reconstructed class, whereas a high error indicates that it is not. To conduct multi-class classification, we are required to create an Autoencoder model for each class and make our selection based on the output with the lowest error.

The design of an Autoencoder is broken into two stages: 1) an Encoder and 2) a Decoder. The Encoder compresses the input while the Decoder decompresses the image to the reconstructed class. Based on our testing, we found that the Autoencoder worked best when Convolutional Layers were used for compressing and decompressing the input. Our Encoder was composed of 2 Convolutional Layers, which decreased the input size to 75% and then 50%. The Decoder was composed of 2 Transpose Convolutional Layers. We modified the size of our Convolutional kernels, the Stride of our Kernels, and the Activation Function during testing. The following describes our model's construction:

1. Encoder:
   a. Convolutional Layer 1 – Kernel: 2 × 2×9, Stride: 2, Activation: ReLU
   b. Convolutional Layer 2 – Kernel: 6 × 6×24, Stride 1, Activation: ReLU
2. Decoder:
   a. Transpose Convolutional Layer 1 – Kernel: 6 × 6×9, Stride: 1, Activation: ReLU
   b. Transpose Convolutional Layer 2 – Kernel: 2 × 2×3, Stride: 1, Activation: ReLU

A visual representation of our Autoencoder model is presented in Figure 7. Additionally, samples of output from the Benign, Kenjiro, and Muhstik Autoencoders are presented in Figure 8.

V. RESULTS AND DISCUSSION

Evaluation of the IoT Botnet Detection Pipeline was conducted using a Jetson Nano and considered the following metrics: 1) Botnet Detection Accuracy (Binary Classification Accuracy), 2) Botnet Classification Accuracy (Multi-Class Classification), 3) Model Throughput in packets/second, and 4) Model Parameter Counts. Before our evaluation, we balanced our datasets to reduce overfitting to the most-represented classes. For the N-BaIoT dataset, we took a random subsample of 10,000 packets for each class from each device, bringing us to 890,000 packets. For the IoT-23 dataset, we took 100,000 samples from each of the 23 scenarios to receive 2,300,000 packets.

After balancing our datasets, we randomly assigned the remaining packets to training, validation, and testing sets with probabilities of 60%, 30%, and 10%, respectively. For our Convolutional Neural Network and Autoencoder, our training, validation, and testing sets were processed via the network traffic visualization methodology. Since the Deep Neural Network serves as our baseline model, no preprocessing was completed. At this stage, the datasets were classified and accuracy metrics were compiled.

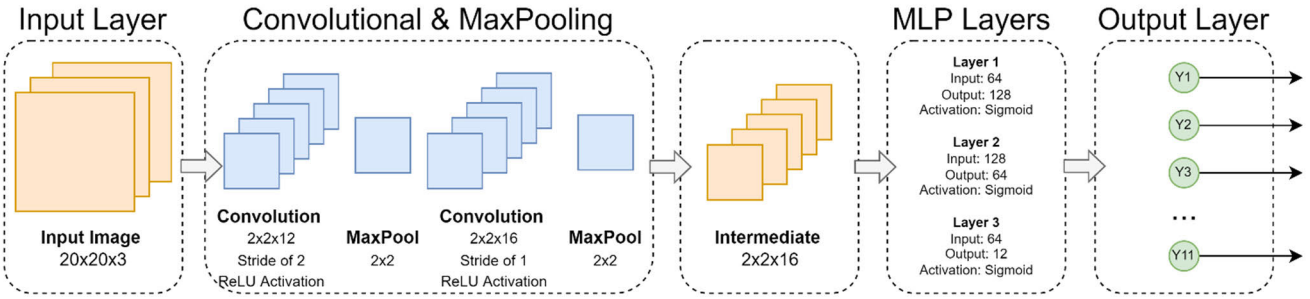


FIGURE 6. The convolutional neural network model used in conjunction with our visualization methodology.

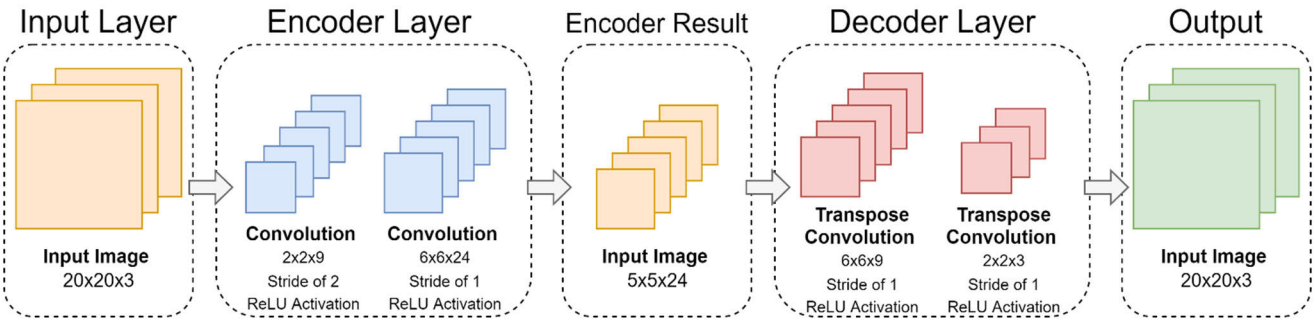


FIGURE 7. The autoencoder model for comparison with the CNN results and the N-BaIoT study.

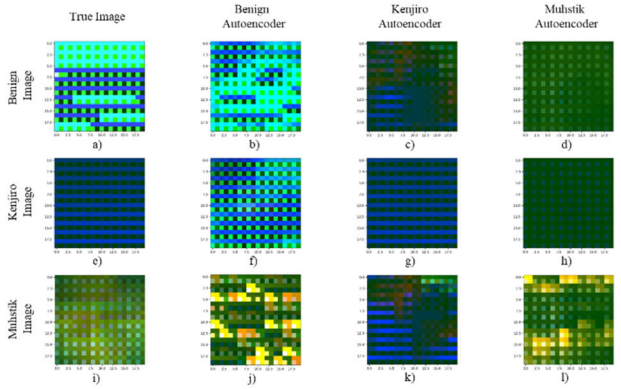


FIGURE 8. Autoencoder results when Benign, Kenjiro, and Muhstik images are provided to the Benign, Kenjiro, and Muhstik Autoencoders.

In subsection V-A., we will present the results for Botnet Detection Accuracy. This metric indicates whether the classifiers detected a botnet, regardless of the type of botnet. Along with detection accuracy, we will present our True Positive Rate, False Positive Rate, True Negative Rate, and False Negative Rate to provide additional insight into model performance. In subsection V-B., we will present the results for Botnet Classification Accuracy. This metric indicates whether our classifiers can identify the Botnet behavior that is present. For the N-BaIoT dataset, 11 classes are present, and for the IoT-23 dataset, 12 classes are present. Subsection V-C. presents the throughput of our IoT Botnet Detection Pipeline and the Deep Neural Network. The throughput of our network traffic visualization methodology and the Convolutional Neural Network will be analyzed as the components of the pipeline. Subsection V-D. presents the Model Parameter Counts for each model, which accounts for the number of weights and biases that are trained for each model. This is an important factor in the storage requirements to host each classifier. Lastly, we will discuss our results in subsection V-E.

A. BOTNET DETECTION ACCURACY

Botnet Detection Accuracy measures how well the proposed pipeline detects whether a botnet is present, regardless of the type of botnet that was active. During operation, our models predicted whether a botnet was present (True) or not (False). Predictions were compared against the observed data, providing us with the True Positive (TP, Botnet observed and predicted), False Positive (FP, Benign traffic observed, but Botnet predicted), True Negative (TN, Benign traffic observed and predicted), and False Negative (FN, Botnet traffic observed, but Benign traffic predicted). Accuracy was calculated according to Equation (1), providing our first metric.

$$Accuracy = \frac{TP + FP}{TP + FP + TN + FP} \tag{1}$$

We applied our Deep Neural Network, Convolutional Neural Network, and Autoencoder models against the N-BaIoT dataset, as seen in Table 3. Based on our results, all three models performed very well at detecting whether botnet activity was taking place. The CNN performed the best with 100%

accuracy, followed closely by the AE at 99.99% accuracy and the DNN at 99.85% accuracy.

TABLE 3. Binary classification metrics – N-BaIoT.

Model	*Accuracy*	*TPR*	*FPR*	*TNR*	*FNR*
DNN	99.85%	99.90%	0.54%	99.46%	0.10%
CNN	**100%**	**100%**	**0%**	**100%**	**0%**
AE	99.99%	100%	0.9%	99.1%	0%

Next, we examined the IoT-23 dataset, receiving the results presented in Table 4. Similar to our N-BaIoT results, all three models had accuracies of >99%, with the Autoencoder performing the best at 99.84%, the Convolutional Neural Network in second at 99.78%, and the Deep Neural network in third with 99.75%.

TABLE 4. Binary classification metrics – IoT-23.

Model	*Accuracy*	*TPR*	*FPR*	*TNR*	*FNR*
DNN	99.75%	99.87%	4.40%	95.60%	0.13%
CNN	99.78%	99.87%	**1.91%**	**98.09%**	0.13%
AE	**99.84%**	**99.98%**	3.10%	96.90%	**0.02%**

To provide further depth to our analysis, we examined the True Positive Rate (TPR, Equation (2)), False Positive Rate (FPR, Equation (3)), True Negative Rate (TNR, Equation (4)), and False Negative Rate (FNR, Equation (5)). These metrics allow us to analyze our misclassifications, indicating whether the models displayed a preference for misclassifying Benign or Botnet behavior. In cybersecurity applications, it is preferred to have a higher False Positive Rater as misclassified Benign traffic may be inspected and cleared, whereas misclassified Botnet traffic will go undetected and cause damage within the network. Across the N-BaIoT and IoT-23 datasets the DNN, CNN, and AE exhibited a greater False Positive Rate, with the DNN presenting the greatest FPR at 4.40%. None of our False Negative Rates exceeded 0.13%, which was an ideal outcome.

$$TPR = \frac{TP}{TP + FN} \tag{2}$$

$$FPR = \frac{FP}{TN + FP} \tag{3}$$

$$TNR = \frac{TN}{TN + FP} \tag{4}$$

$$FNR = \frac{FN}{TP + FN} \tag{5}$$

B. BOTNET CLASSIFICATION ACCURACY

Next, we expanded our analysis to the accuracy of the models when considering the type of attack launched in the N-BaIoT and IoT-23 datasets. Since N-BaIoT is comprised of 10 sub-classifications of Mirai and Gafgyt activity, we considered the classification of the overall group along with the individual attack characteristics. Beginning with the general classes, we observed high accuracy for all three models. The highest accuracy was the Convolutional Neural Network at 99.83% accuracy, followed by the Deep Neural Network at 99.57%, and then the Autoencoder at 97.87%. As we can see, the DNN and CNN models continued to perform well when our classification was expanded; however, the AE model experienced a dip in accuracy. This trend will continue as we expand our classification to the sub-classes and shows a weakness in the AE model when expanding past binary classification. To further explore the results, we calculated the confusion matrices for each model, as shown in Figures 9 (Deep Neural Network), 10 (Convolutional Neural Network), and 11 (Autoencoder). When examining the misclassifications of the Autoencoder, we see that the model had a greater challenge when correctly classifying the Gafgyt data, often mistaking it for Mirai data.

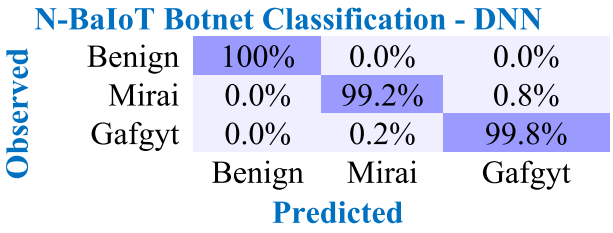


FIGURE 9. Confusion matrix for botnet classification of the N-BaIoT dataset using the deep neural network.

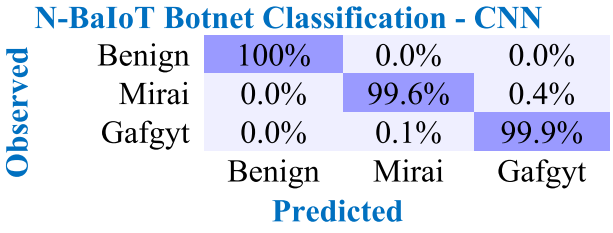


FIGURE 10. Confusion matrix for botnet classification of the N-BaIoT dataset using the convolutional neural network.

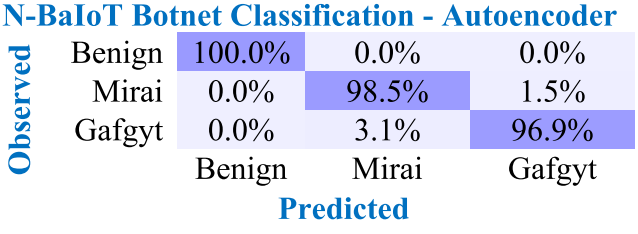


FIGURE 11. Confusion matrix for botnet classification of the N-BaIoT dataset using the autoencoder.

Following our general classification of Mirai and Gafgyt activity within the N-BaIoT data, we studied our models' accuracy for detecting the sub-classifications. We found that the classification accuracy dropped considerably for the 11 classes (1 benign, 5 Mirai, and 5 Gafgyt) for all three models. Among the three models, the Deep Neural Network achieved the highest accuracy at 78.74%, followed by the Convolutional Neural Network at 69.29% and the Autoencoder at 66.87%. To better understand our CNN model's

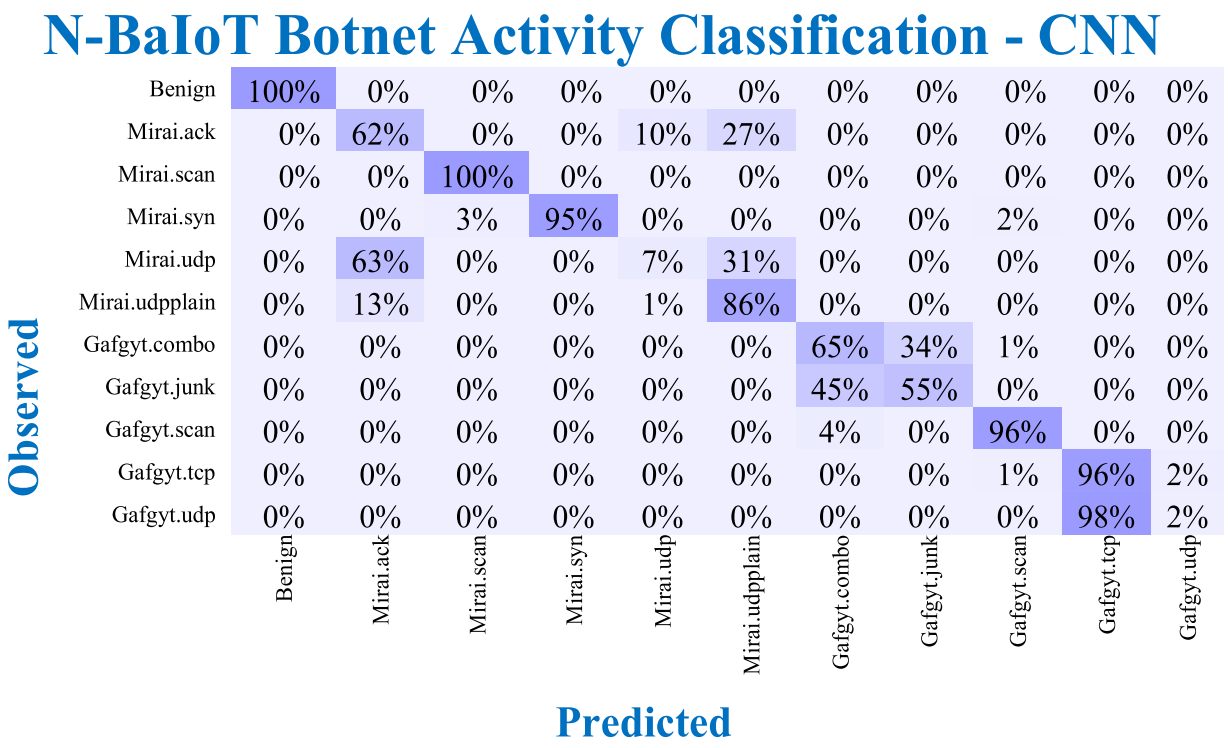


FIGURE 12. Confusion matrix for Mirai and Gafgyt sub-classification of the N-BaIoT dataset using the convolutional neural network.

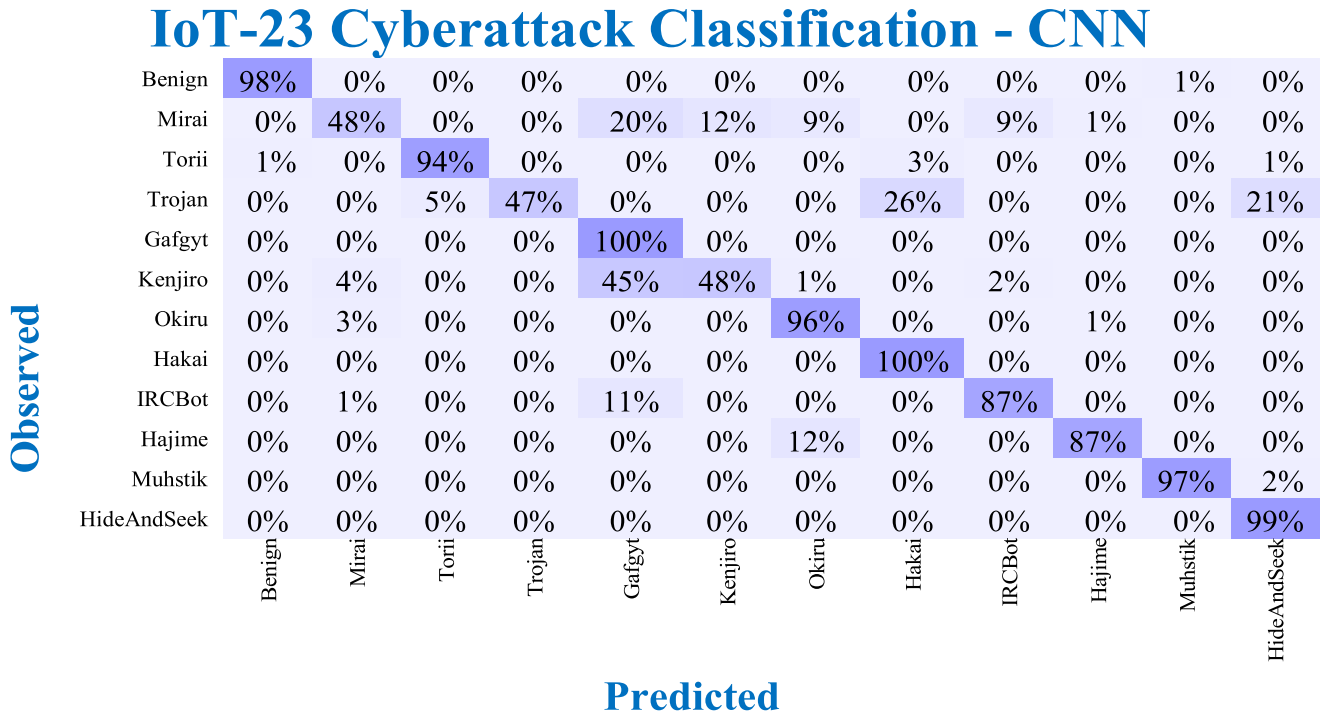


FIGURE 13. Confusion matrix for cyberattack classification of the IoT-23 dataset using the convolutional neural network.

drastic drop in accuracy, we present its confusion matrix in Figure 12. From the confusion matrix, we see that the model had difficulty in differentiating between the Mirai UDP and Mirai ACK activity, the Gafgyt UDP and the Gafgyt TCP activity, and the Gafgyt Combo and the Gafgyt Junk activity. By taking a deeper look at the visualized sample data in Figure 2, we note that these sub-classes were visually similar and are the most likely cause of these misclassifications.

Lastly, we examined the classification accuracy when working with the IoT-23 Dataset. Regarding the overall accuracy, the Deep Neural Network achieved an accuracy of 72.22%, the Convolutional Neural Network achieved an overall accuracy of 76.31%, and the Autoencoder achieved an accuracy of 66.80%. In this situation, the visualization methodology performed better than the traditional approach; however, it still stumbled when applied under the Autoencoder. Examining the confusion matrix of the CNN model Figure 13, we note that the model had difficulty distinguishing between the Mirai, Gafgyt, and Kenjiro botnets. This was an unexpected result as the model had performed very well when comparing the Mirai and Gafgyt data from the N-BaIoT dataset. A possible explanation is that since Mirai and Gafgyt are from the same family, the diversity of botnets in IoT-23 caused the differences to become less notable during classification.

C. THROUGHPUT

After considering each model's detection and classification accuracy on the N-BaIoT and IoT-23 datasets, we calculated their throughput on a Jetson Nano. When processed by our proposed methodology, packets will undergo a three-stage process. First, they will be collected by the Jetson Nano. By setting the Jetson Nano as the local Access Point/Router, traffic will automatically be collected as it flows between the local and external networks, having a negligible impact on throughput. Second, packets will be passed through the network traffic visualization procedure. During testing, this required an average of 35 microseconds per packet, equating to a throughput of roughly 28,500 packets per second. Lastly, each image is classified by the Convolutional Neural Network.

For a comparison with botnet detection without the proposed pipeline, we offer throughput for the Deep Neural Network and Convolutional Neural Network in Table 5. Using a batch size of 64 packets, the Deep Neural Network achieved a throughput of 7000 packets per second. We observed a throughput of 17,000 packets per second for our Convolutional Neural Network. This translates to a roughly 2.4 times greater throughput of our CNN versus the DNN model.

TABLE 5. Classification throughput.

Model	*Throughput (packets/second)*
DNN	7,000
CNN	**17,000**

Combined, these results show that the proposed pipeline takes 95 microseconds to process and classify a single packet. However, during steady-state operation, the maximum performance of the pipeline is measured by the worst performance of the three stages, which, in this case, is our CNN classifier. As a result, the observed throughput of the pipeline is 17,000 packets per second.

We only included results for our Deep Neural Network and Convolutional Neural Network for our analysis. To complete classification with the autoencoder model, we needed a generator and classifier for each class. Due to this architecture, the models could be run sequentially or in parallel, leading to greater inconsistencies in the overall throughput of the AE model.

The proposed methodology was designed for deployment on the Jetson Nano, which can process up to 472 gigaflops while maintaining the cost and form factor of a high-end embedded device. This supplies ample processing power at a lower cost compared to standalone servers with GPUs. For applications where the Jetson Nano would be too expensive, the methodology may be deployed on a Raspberry Pi 3, which can process up to 12 gigaflops. We expect this to drop our CNN throughput to roughly 510 packets per second. If we consider the IoT-23 dataset, we saw an overall average of 436.85 packets per second when monitoring a 3-device network. Based on these results, both the Jetson Nano and Raspberry Pi are sufficient, with the Raspberry Pi struggling when exposed to higher loads. Additional testing is required for more expansive IoT networks, and results may differ depending on the type of IoT devices present in the environment.

D. PARAMETER COUNTS

Our last objective was to evaluate the number of parameters trained for the model. Due to the lightweight design of IoT devices, storage is limited, and our machine learning models should be accurate while maintaining a small footprint. A comparison of parameter counts between our three models is presented in Table 6, with the Deep Neural Network composed of 23,207 trainable parameters, the Convolutional Neural Network composed of 18,231 trainable parameters, and the Autoencoder composed of 189,756 trainable parameters. Based on these results, the visualization methodology allowed us to decrease the number of parameters by 21.4%. The Autoencoder was high for this approach as we created a model for each cyberattack class. If we had only focused on binary classification, we would have seen a parameter count of 15,812, less than what was needed for our CNN model.

TABLE 6. Machine learning model parameter counts.

Model	*Parameter Count*
DNN	23,207
CNN	**18,231**
AE	189,756

E. DISCUSSION

Based on our results, our proposed IoT Botnet Detection Pipeline achieves very high botnet detection accuracies when applied to two independent datasets, the N-BaIoT and IoT-23 datasets. A summary of our cyberattack detection and multi-class classification is presented in Table 7. When

the visualization pipeline is applied towards binary classification, it achieved 100% accuracy against the N-BaIoT dataset using the Convolutional Neural Network and 99.84% against the IoT-23 dataset using an Autoencoder. Similarly, the technique was successful at classifying between general Mirai and Gafgyt activity, with the CNN providing the best accuracy at 99.83%. While we were successful in botnet detection, substantial improvement is required in achieving greater multi-class classification, as our models could not achieve accuracy greater than 78%.

TABLE 7. Classification accuracy.

Model	*N-BaIoT*			*IoT-23*	
	Binary Class.	Mirai v. Gafgyt	Multi-Class.	Binary Class.	Multi-Class.
DNN	99.85%	99.57%	**78.74%**	99.75%	72.22%
CNN	**100%**	**99.83%**	69.29%	99.78%	**76.31%**
AE	99.99%	97.87%	66.87%	**99.84%**	66.80%

Upon further investigation of our multi-class classification results for our IoT-23 datasets, we see that the methodology experienced difficulty identifying the Trojan and Kenjiro botnets. If we examine visualized samples of the observed Trojan activity and model's predictions in Figure 14, we observe that their traffic displays very similar characteristics. This highlights a limitation in our methodology in identifying activity with similar behavior. A potential solution to this challenge may be to increase the size and number of layers for our Convolutional Neural Network; however, this will have an adverse impact on our performance. Alternatively, we may pivot from botnet classification to classifying the stage of infection. For instance, we may classify activity related to reconnaissance, attempts at vulnerability exploitation, indicators of command-and-control functionality, and denial of service. These classes would offer a stronger indication of the severity or stage of the cyberattack to the detection engineer.

Despite struggling with multi-class classification, our models excelled at binary classification, and we are confident that these results will hold when exposed to new environments. When searching for our datasets, we based our selection on the diversity of consumer IoT devices and botnets. The N-BaIoT dataset studied 9 consumer IoT devices during the propagation and attack stage of the Mirai and Gafgyt botnets. Since only 2 Botnets were investigated, we expanded our investigation to a second dataset, the IoT-23 dataset. The IoT-23 dataset examined the effects of 11 botnets against 3 devices. Since our methodology was equally effective in detecting malicious activity across both datasets, we are confident that these results will hold against other IoT environments.

Compared to our Deep Neural Network, which did not utilize the visualization methodology, our Convolutional Neural Network achieved better accuracy, throughput, and parameter counts during binary classification of both datasets. Further, we achieved high accuracy while only using 12 features compared to the 115 used by the DNN model. We considered

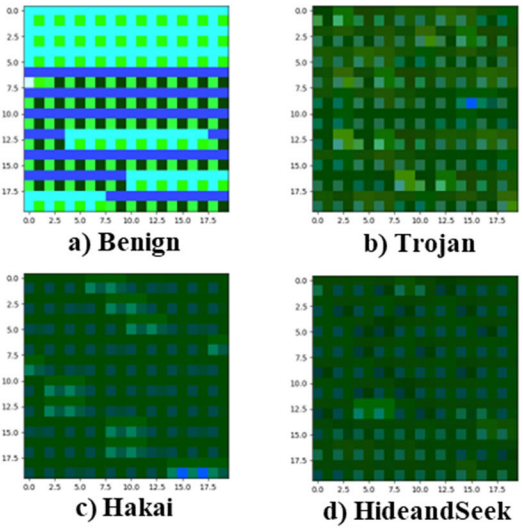


FIGURE 14. Visualized network traffic of Benign, Trojan, Hakai, and HideandSeek activity from the IoT-23 dataset.

these findings important as the DNN is the most straightforward Deep Learning algorithm, showing the flexibility of the pipeline approach in lightweight cyberattack detection for IoT devices.

Additionally, our expansion of the Autoencoder model performed well in binary classification within the N-BaIoT and IoT-23 datasets. Our model differed from the model proposed by the N-BaIoT authors, as ours was designed for device-agnostic attack detection. In contrast, the N-BaIoT authors trained a model for each device within the environment. However, additional work needs to be done to improve multi-class classification.

VI. CONCLUSION

Through our research, we evaluated a novel network traffic visualization methodology for botnet detection and classification within an Internet of Things environment. Coupled with a Convolutional Neural Network, we showed that the methodology achieved >98% accuracy across the N-BaIoT and IoT-23 datasets. Further, compared to a baseline Deep Neural Network model, we achieved comparable accuracy while reducing the parameter count by 21.4% and achieving a 2.4 times greater throughput. However, we found that the model struggled to achieve a classification accuracy greater than 76% when expanding the classes to the specific cyberattack.

In addition to showing proficiency in automated botnet detection, the network traffic visualization methodology may be integrated into the existing workflow for cyberattack detection engineers. Based on our sample images presented in Figures 3 and 4 (with a sub-sample in Figure 14), we can observe clear trends when comparing the Benign and Botnet traffic. By integrating these images into engineering dashboards, personnel can receive a visual indicator of the general health of their network. Shifts in the color or pattern of

the visualized traffic may indicate an ongoing cyberattack. By identifying these shifts, staff may catch the cyberattack and initiate incident response procedures in a timely manner.

While we were successful in our objective of creating a lightweight IoT Botnet Detection Pipeline, there are many avenues for future exploration. First, highly skilled threat actors go to extensive lengths to evade detection. To evade our proposed solution, the attacker's traffic would need to present similar characteristics to Benign network traffic. This could be easily accomplished during reconnaissance as the attacker could present themselves as a regular user; however, this becomes more challenging as the attacker progresses through their cyber kill chain, exploiting vulnerabilities and establishing command-and-control functionality. In our future research, we'll determine whether easily masked activity, such as reconnaissance and scanning, can be identified by our methodology.

Additionally, evaluation of the IoT Botnet Detection Pipeline's adaptability to novel attack chains is an ongoing effort. Since the evolution of existing threats can occur in any direction, it's difficult to assess how our pipeline would react to these changes. Potential solutions to this challenge include expanding our feature selection to the payload of the network traffic. When designing the network traffic visualization methodology, it was desirable to focus on the packet's metadata to speed up image generation. However, the payload may contain critical information that may reveal how the adversary exploited the target device. Identifying methods for interpreting the payload is a core research direction for identifying these unseen threats.

REFERENCES

[1] S. H. Shah and I. Yaqoob, "A survey: Internet of Things (IoT) technologies, applications and challenges," *IEEE Smart Energy Grid Engineering (SEGE)*, Aug./Oct. 2016, pp. 381–385.

[2] S. Balaji, K. Nathani, and R. Santhakumar, "IoT technology, applications and challenges: A contemporary survey," *Wireless Pers. Commun.*, vol. 108, pp. 363–388, Jun. 2019.

[3] S. Sinha. (2023). *State of IoT 2023: Number of Connected IoT Devices Growing 16% To 16.7 Billion Globally*. Accessed: Nov. 22, 2023. [Online]. Available: https://iot-analytics.com/number-connected-iot-devices/

[4] *Cisco Annual Internet Report (2018–2023)*, Cisco, San Jose, CA, USA, 2018.

[5] P. I. Radoglou Grammatikis, P. G. Sarigiannidis, and I. D. Moscholios, "Securing the Internet of Things: Challenges, threats and solutions," *Internet Things*, vol. 5, pp. 41–70, Mar. 2019.

[6] M. Serror, S. Hack, M. Henze, M. Schuba, and K. Wehrle, "Challenges and opportunities in securing the industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 2985–2996, May 2021.

[7] A. R. Sfar, E. Natalizio, Y. Challal, and Z. Chtourou, "A roadmap for security challenges in the Internet of Things," *Digit. Commun. Netw.*, vol. 4, no. 2, pp. 118–137, 2018.

[8] H. Taherdoost, "Security and Internet of Things: Benefits, challenges, and future perspectives," *Electron.*, vol. 12, no. 8, pp. 1901–1922, Sep. 2023.

[9] S. Ahmed and M. Khan, "Securing the Internet of Things (IoT): A comprehensive study on the intersection of cybersecurity, privacy, and connectivity in the IoT ecosystem," *AI, IoT Fourth Industrial Revolution Review*, vol. 13, no. 9, pp. 1–17, 2023.

[10] (2018). *Internet of Things (IoT) Top 10 2018*. Accessed: Nov. 20, 2023. [Online]. Available: https://wiki.owasp.org/index.php/OWASP_Internet_of_Things_Project

[11] M. Gromov, D. Arnold, and J. Saniie, "Tackling multiple security threats in an IoT environment," in *Proc. IEEE Int. Conf. Electro Inf. Technol.*, May 2022, pp. 290–295.

[12] N. Neshenko, E. Bou-Harb, J. Crichigno, G. Kaddoum, and N. Ghani, "Demystifying IoT security: An exhaustive survey on IoT vulnerabilities and a first empirical look on Internet-scale IoT exploitations," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 3, pp. 2702–2733, 3rd Quart., 2019.

[13] Z. Mohammad, T. A. Qattam, and K. Saleh, "Security weaknesses and attacks on the Internet of Things applications," in *Proc. IEEE Jordan Int. Joint Conf. Electr. Eng. Inf. Technol. (JEEIT)*, Apr. 2019, pp. 431–436.

[14] M. Abomhara and G. M. Koien, "Cyber security and the Internet of Things: Vulnerabilities, threats, intruders and attacks," *J. Cyber Secur. Mobility*, vol. 4, pp. 65–88, May 2015.

[15] U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, "A critical cybersecurity analysis and future research directions for the Internet of Things: A comprehensive review," *Sensors*, vol. 23, no. 8, p. 4117, Apr. 2023.

[16] I. Ahmad, M. S. Niazy, R. A. Ziar, and S. Khan, "Survey on IoT: Security threats and applications," *J. Robot. Control*, vol. 2, no. 1, pp. 42–46, 2021.

[17] M. Antonakakis, T. April, B. Michael, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, D. Kumar, C. Lever, Z. Ma, J. Mason, and Y. Zhou, "Understanding the mirai botnet," in *Proc. 26th USENIX Security Symp.*, 2017, pp. 1–18.

[18] A. Marzano, D. Alexander, O. Fonseca, E. Fazzion, C. Hoepers, K. Steding-Jessen, M. H. P. C. Chaves, Ì. Cunha, D. Guedes, and W. Meira, "The evolution of bashlite and mirai IoT botnets," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 00813–00818.

[19] L. McNulty and V. G. Vassilakis, "IoT botnets: Characteristics, exploits, attac capabilities, and targets," in *Proc. 13th Int. Symp. Commun. Syst.*, 2022, pp. 1–18.

[20] C. Lin. (2023). *IZ1H9 Compain Enhances its Arsenal With Scores of Exploits*. Accessed: Nov. 22, 2023. [Online]. Available: https://www.fortinet.com/blog/threat-research/Iz1h9-campaign-enhances-arsenal-with-scores-of-exploits

[21] S. Dange and M. Chatterjee, "IoT botnet: The largest threat to the IoT network," in *Proc. GUCON*, 2019, pp. 1–18.

[22] M. Zang, C. Zheng, L. Dittmann, and N. Zilberman, "Towards continuous threat defense: In-network traffic analysis for IoT gateways," *IEEE Internet Things J.*, vol. 11, no. 1, pp. 9244–9257, Aug. 2023.

[23] M. Nankya, R. Chataut, and R. Akl, "Securing industrial control systems: Components, cyber threats, and machine learning-driven defense strategies," *Sensors*, vol. 23, no. 21, p. 8840, Oct. 2023.

[24] M. Manimaran, M. Dhar, R. Norabuena-Figueroa, M. R. S. Saraswathi, and K. Selvakumarasamy, "Implementing machine learning-based autonomic cyber defense for IoT-enabled healthcare devices," *J. Artif. Intell. Technol.*, vol. 3, no. 4, pp. 162–172, Jun. 2023.

[25] J. Bhayo, S. A. Shah, S. Hameed, A. Ahmed, J. Nasir, and D. Draheim, "Towards a machine learning-based framework for DDOS attack detection in software-defined IoT (SD-IoT) networks," *Eng. Appl. Artif. Intell.*, vol. 123, Aug. 2023, Art. no. 106432.

[26] M. W. Nadeem, H. G. Goh, Y. Aun, and V. Ponnusamy, "Detecting and mitigating botnet attacks in software-defined networks using deep learning techniques," *IEEE Access*, vol. 11, pp. 49153–49171, 2023.

[27] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-BaIoT—network-based detection of IoT botnet attacks using deep autoencoders," *IEEE Pervasive Comput.*, vol. 17, no. 3, pp. 12–22, Jul. 2018.

[28] S. Garcia, A. Parmisano, and M. J. Erquiaga. (2020). *IoT-23: A Labeled Dataset With Malicious and Benign IoT Network Traffic (Version 1.0.0)*. [Online]. Available: https://www.stratosphereips.org/datasets-iot23

[29] C. Frank, S. Jarocki, C. Nance, and W. E. Pauli, "Protecting IoT devices from the mirai botnet," *J. Inf. Syst. Appl. Res.*, vol. 11, no. 2, pp. 33–44, 2018.

[30] C. Dietz, R. L. Castro, J. Steinberger, C. Wilczak, M. Antzek, A. Sperotto, and A. Pras, "IoT-botnet detection and isolation by access routers," in *Proc. 9th Int. Conf. Netw. Future*, Nov. 2018, pp. 88–95.

[31] R. Chaganti, B. Bhushan, and V. Ravi, "A survey on blockchain solutions in DDoS attacks mitigation: Techniques, open challenges and future directions," *Comput. Commun.*, vol. 197, pp. 96–112, May 2023.

[32] Q. Shafi and A. Basit, "DDoS botnet prevention using blockchain in software defined Internet of Things," in *Proc. 16th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Jan. 2019, pp. 624–628.

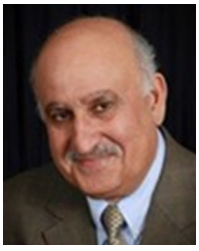
[33] Z. Ahmed, S. M. Danish, H. K. Qureshi, and M. Lestas, "Protecting IoTs from mirai botnet attacks using blockchains," in *Proc. IEEE 24th Int. Workshop Comput. Aided Model. Design Commun. Links Netw. (CAMAD)*, Sep. 2019, pp. 1–6.

[34] Z. E. Huma, S. Latif, J. Ahmad, Z. Idrees, A. Ibrar, Z. Zou, F. Alqahtani, and F. Baothman, "A hybrid deep random neural network for cyberattack detection in the industrial Internet of Things," *IEEE Access*, vol. 9, pp. 55595–55605, 2021.

[35] F. Hussain, S. G. Abbas, I. M. Pires, S. Tanveer, U. U. Fayyaz, N. M. Garcia, G. A. Shah, and F. Shahzad, "A two-fold machine learning approach to prevent and detect IoT botnet attacks," *IEEE Access*, vol. 9, pp. 163412–163430, 2021.
[36] I. Ullah and Q. H. Mahmoud, "Design and development of a deep learning-based model for anomaly detection in IoT networks," *IEEE Access*, vol. 9, pp. 103906–103926, 2021.
[37] B. I. Hairab, M. Said Elsayed, A. D. Jurcut, and M. A. Azer, "Anomaly detection based on CNN and regularization techniques against zero-day attacks in IoT networks," *IEEE Access*, vol. 10, pp. 98427–98440, 2022.
[38] N. Abdalgawad, A. Sajun, Y. Kaddoura, I. A. Zualkernan, and F. Aloul, "Generative deep learning to detect cyberattacks for the IoT-23 dataset," *IEEE Access*, vol. 10, pp. 6430–6441, 2022.
[39] H. Alkahtani and T. H. H. Aldhyani, "Botnet attack detection by using CNN-LSTM model for Internet of Things applications," *Secur. Commun. Netw.*, vol. 2021, pp. 1–23, Sep. 2021.
[40] M. Saharkhizan, A. Azmoodeh, A. Dehghantanha, K. R. Choo, and R. M. Parizi, "An ensemble of deep recurrent neural networks for detecting IoT cyber attacks using network traffic," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8852–8859, Sep. 2020.
[41] F. Sattari, A. H. Farooqi, Z. Qadir, B. Raza, H. Nazari, and M. Almutiry, "A hybrid deep learning approach for bottleneck detection in IoT," *IEEE Access*, vol. 10, pp. 77039–77053, 2022.
[42] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2018, pp. 664–669.
[43] Y. He and W. Li, "Image-based encrypted traffic classification with convolution neural networks," in *Proc. IEEE 5th Int. Conf. Data Sci. Cyberspace (DSC)*, Jul. 2020, pp. 271–278.
[44] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis, "IoT malware network traffic classification using visual representation and deep learning," in *Proc. 6th IEEE Conf. Netw. Softwarization*, Jun. 2020, pp. 444–449.
[45] S. Kalenowski, D. Arnold, M. Gromov, and J. Saniie, "Heterogeneity tolerance in IoT botnet attack classification," in *Proc. IEEE Int. Conf. Electro Inf. Technol.*, May/Jul. 2023, pp. 353–356.
[46] A. Agniel, D. Arnold, and J. Saniie, "Image processing for detecting botnet attacks: A novel approach for flexibility and scalability," in *Proc. IEEE Int. Conf. Expo. Real Time Commun.*, Oct. 2022, pp. 8–12.
[47] M. Gromov, D. Arnold, and J. Saniie, "Edge computing for real time botnet propagation detection," in *Proc. IEEE Int. Conf. Expo. Real Time Commun.*, Oct. 2022, pp. 13–16.
[48] M. Gromov, D. Arnold, and J. Saniie, "Utilizing computer vision algorithms to detect and classify cyberattacks in IoT environments in real-time," in *Proc. IEEE Int. Conf. Electro Inf. Technol.*, May 2023, pp. 1–18.

DAVID ARNOLD (Member, IEEE) received the B.S. degree in electrical engineering and the M.S. degree in computer engineering from Illinois Institute of Technology, Chicago, IL, USA, in 2019, where he is currently pursuing the Ph.D. degree in computer engineering. He is also a Cybersecurity Researcher with the Embedded Computing and Signal Processing (ECASP) Research Laboratory, Illinois Institute of Technology. His research interests include the Internet of Things, machine learning, homomorphic encryption, and cybersecurity testbeds. He was a recipient of the Grainger Fellowship. He received the University Nuclear Leadership Program's Graduate Fellowship, in 2020, supporting his research into cyber-secure network architectures for nuclear power plants. He founded Illinois Tech CyberHawks, a cybersecurity student organization focused on enhancing cybersecurity education. He also instructs ECE 222, introduction to cybersecurity engineering.

MIKHAIL GROMOV (Member, IEEE) received the B.S. degree in computer and cybersecurity engineering and the M.S. degree in computer engineering from Illinois Institute of Technology, Chicago, IL, USA, where he is currently pursuing the Ph.D. degree in computer engineering. He is also a Researcher with the Embedded Computing and Signal Processing (ECASP) Research Laboratory, Illinois Institute of Technology. His main research interests include cybersecurity, machine learning, and the IoT, along with interests in many other fields, including FPGA development and signal processing. He was a member of the CyberHawks Executive Board, which is the cybersecurity organization at IIT.



JAFAR SANIIE (Life Fellow, IEEE) received the B.S. degree (Hons.) in electrical engineering from the University of Maryland, College Park, MD, USA, in 1974, the M.S. degree in biomedical engineering from Case Western Reserve University, Cleveland, OH, USA, in 1977, and the Ph.D. degree in electrical engineering from Purdue University, West Lafayette, IN, USA, in 1981. In 1981, he joined the Department of Applied Physics, University of Helsinki, Helsinki, Finland, to conduct research in photothermal and photoacoustic imaging. Since 1983, he has been with the Department of Electrical and Computer Engineering, Illinois Institute of Technology, Chicago, IL, USA, where he is the Department Chair, the Filmer Endowed Chair Professor, and the Director of the Embedded Computing and Signal Processing (ECASP) Research Laboratory. He has over 370 publications and has supervised 35 Ph.D. dissertations and 22 M.S. theses to completion. His research interests and activities are in ultrasonic signal and image processing, ultrasonic software-defined communications, artificial intelligence and machine learning, statistical pattern recognition, estimation and detection, data compression, time-frequency analysis, embedded digital systems, system-on-chip hardware/software codesign, the Internet of Things and cybersecurity, computer vision, deep learning, and ultrasonic nondestructive testing and imaging. He is an IEEE Life Fellow for contributions to ultrasonic signal processing for detection, estimation, and imaging. Since 1987, he has been a Technical Program Committee Member of the IEEE Ultrasonics Symposium. He received the 2007 University (Illinois Institute of Technology) Excellence in Teaching Award. He was the Chair of the Sensors, Nondestructive Evaluation (NDE), and Industrial Applications Group, from 2004 to 2013, the Publication Chair of the IEEE Ultrasonics Symposium, in 2013 and 2021, and the General Chair of the 2014 IEEE Ultrasonics Symposium in Chicago. Since 2018, he has been the IEEE UFFC Awards Chair. From 2014 to 2017, he served as the Ultrasonics Vice President for the IEEE Ultrasonics, Ferroelectrics, and Frequency Control (UFFC) Society. He was a Lead Guest Editor for the Special Issue on Ultrasonics and Ferroelectrics of IEEE TRANSACTIONS ON ULTRASONICS, FERROELECTRICS, AND FREQUENCY CONTROL, in August 2014, the Special Issue on Novel Embedded Systems for Ultrasonic Imaging and Signal Processing of IEEE TRANSACTIONS ON ULTRASONICS, FERROELECTRICS, AND FREQUENCY CONTROL in July 2012, and the Special Issue on Advances in Acoustic Sensing, Imaging, and Signal Processing of *Journal of Advances in Acoustics and Vibration* in 2013. Between 1994 and 2020, he was an Associate Editor of IEEE TRANSACTIONS ON ULTRASONICS, FERROELECTRICS, AND FREQUENCY CONTROL.