MetaIoT: Few Shot Malicious Traffic Detection in Internet of Things Networks Based on HIN

Hongwu Li[1,2] , Xingyu Fu*[1,2], Yujia Zhu[1,2], Rong Yang[1,2], Chao Li[3]
[1]*Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China*
[2]*School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China*
[3]*National Computer network Emergency Response technical Team/Coordination Center*
Corresponding author: fuxingyu@iie.ac.cn

***Abstract*—Identification of abnormal and malicious traffic in the Internet-of-Things (IoT) network is critical for IoT security. However, it is worth noting that the majority of recent efforts demand a large amount of tagged traffic to train a machine-learning model. In this paper, we develop MetaIoT, an intelligent approach for identifying malicious traffic. MetaIoT is more accurate and more difficult for attackers to circumvent by taking into account both the local attributes of each traffic source and their global relationships. In MetaIoT, we begin by considering the heterogeneous and dynamic nature of traffic. Then, we introduce a heterogeneous graph (HG) to model the relationships between traffic and employ a relation-based heterogeneous graph attention network to learn node (i.e., traffic) representations over the built HG. Alternatively, MetaIoT addresses the issue of needing enough data for model training through the meta-learning technique. After conducting a comprehensive comparison with the baseline through experiments, our model demonstrated superior performance in few-shot learning scenarios, obtaining an accuracy score of 91.65% and an F1 score of 90.33%. When compared with current state-of-the-art IoT traffic detection models, our model showed the best results.**

***Index Terms*—IOT network security, meta-learning, heterogeneous information network, graph neural networks**

I. Introduction

International Data Corporation (IDC) estimates that there will be 41.6 billion connected IoT devices in 2025 [1]. The IoT security situation is worsening with adversaries creating malicious network traffic, resulting in device corruption, DoS attacks, and malware installations. One such attack was launched in 2016, famously known as the Mirai attack [2]. Identifying malicious attacks by detecting traffic is one of the easiest and most effective defences.

In the literature, malicious IoT traffic detection is performed by first extracting statistical and behavioral network traffic features [3], and then using machine learning methods, such as LDA, RF, CART, SVM, LSTM, etc. These studies construct classifiers using attributes taken from malicious Internet of Things (IoT) traffic, logs, or other enrichment data. Because they deal with each region separately and rely on manual features, feature-based systems can be readily evaded by well-designed attacks from smart opponents, even when they attain high performance in their ratings. To tackle this issue, researchers suggest a neural network-based learning algorithm that can extract deep harmful flow characteristics [4], [5]. In order to recognize anomalous behaviors, Yujia Li et al. [6] created a Gated Recurrent Neural Network (GRNN), whose output is sent into the detector engine. For the security of IoT networks, Khan et al. [7] proposed a unique bidirectional SRU-driven DL model leveraging skip connections. However, some of these systems can't properly uncover the intricate linkages between traffic because of the limitations of these neural network models, leaving significant information untouched.

The graph structure is a model that can effectively depict the relationship mentioned above, but conventional machine learning algorithms place too much emphasis on attributes, making it difficult for them to comprehend graphical data. Known as the Heterogeneous Information Network (HIN) [8], the real-world network often has a variety of nodes and edges. It follows that malicious traffic might be distinguished from HIN. In this research, we shall refer to such intricate networked data without distinction using the terms 'HIN' and 'HG' (heterogeneous graph). Because the heterogeneous graph has richer semantics and more extensive information, we utilize it as the input of our model. The structure of metapath [9], which is a composite relation between two objects, is frequently utilized to understand these semantics. For such graphical data, the Graphical Neuronal Network (GNN), a potent deep-representation learning technique, has demonstrated improved performance in network analysis. According to [10], by taking into account the attention mechanism in heterogeneous graph learning, it is possible to successfully learn information from numerous meta-path specified connections.

Furthermore, the majority of recent efforts demand enough labeled data to train the model for the detection of malicious internet of things traffic. For instance, [11]–[13] develop the IDS system using the entire Ton_IoT dataset [14]. Obtaining labeled samples in the real world is always expensive, and existing algorithms may struggle to detect fraudulent traffic due to the lack of labeled data.

In this paper, we offer a comprehensive system called MetaIoT (as illustrated in Figure 3) to automatically detect malicious IoT traffic in smart settings, in order to overcome the aforementioned issues. Initially, we construct a heterogeneous graph based on traffic features. The created HG is capable of thoroughly characterizing the intricate IoT traffic ecosystem. Then, we employ a relation-based heterogeneous Graph Attention Network (HAN) [8] to combine relational

ISBN 978-3-903176-57-7

information about entities and to get the first node embeddings over the constructed HG. Besides, we suggest a meta-learning architecture to efficiently adapt knowledge from training tasks to testing tasks in order to address the problem of few labeled samples (e.g., new types of malicious traffic with few labeled samples). MetaIoT is trained and optimized through several comparable few-shot learning tasks to improve generalization of learning new tasks as opposed to existing popular GNNs-based models that just depend on the auxiliary information of nodes and aggregated information from neighbors. To summarize, the following are the primary contributions of our work:

- In a realistic scenario, the proportion of attack traffic is small, therefore we adjust the dataset such that malicious traffic only accounts for 3% of normal traffic to simulate the traffic distribution in real-life scenarios. Furthermore, based on the dataset, we analyze the characteristics of the attacker's resource concentration and the distribution difference of commonly used ports for different types of traffic. Subsequently, we have constructed an heterogeneous graph representation of the IoT traffic, utilizing these findings.
- Considering the dynamic and heterogeneous nature of most real-world IoT traffic, we introduce a scalable and heterogeneous GCN module. MetaIoT can support inductive learning in HIN and handle its node features and graph structure information simultaneously, fully excavating the relationship between traffic and transferring it to the node embedding, using meta-path guided short walks and attention-based aggregations.
- We propose a new graph few-shot learning system for malicious IoT traffic in response to the problem of scarcely labeled traffic data.It differs from previous work in that we intend to classify malicious nodes into new classes using only a few samples.
- Comprehensive experiments compared with the baseline and several state-of-the-art IoT traffic detection models demonstrate the outstanding performance of MetaIoT. As IoT security attack events have continued to increase over the past decade across the country, our proposed technique will have a significant societal impact to help address this critical issue.

II. Background and related work

In this section, we first introduce backgrounds about Ton_IoT dataset [14]. Then we recap the basic concepts in heterogeneous graph attention network and Meta-learning as MetaIoT mainly relies on these two models.

A. Ton_IoT Dataset

Ton_IoT is a collection of data sets that are a new generation of IoT and Industrial IoT (IIoT) data sets for evaluating the fidelity and efficiency of different cybersecurity applications based on artificial intelligence (AI) [14]. They are targeted towards applications including adversarial machine learning, intrusion detection, threat intelligence, and privacy-preserving models. The data sets were given the name "Ton_IoT" since they were compiled from a variety of sources, including data sets from network traffic, Windows 7 and 10 operating systems, as well as Ubuntu 14 and 18. The Ton_IoT dataset contains threats related to backdoors, DDoS, DoS, scanning, injection, ransomware, man-in-the-middle (MitM), cross-site scripting (XSS), and password attacks, and consists of 300,000 normal and 161,043 threat observations.

B. Heterogeneous Information Network

Recently, there has been more attention on the differences between various components and linkages in real-world systems, instead of treating them as uniform networks. Heterogeneous Information Networks (HINs) contain multiple types of nodes and edges, each with unique features. HINs are commonly used in data mining because they provide more information and rich semantics. Relationships between nodes are represented by "meta-paths" which can vary in meaning.

C. Heterogeneous Graph Attention Network

Graph neural networks are a powerful approach for graph representation in deep learning and have shown great results. However, they face challenges when applied to heterogeneous graphs with rich semantic information. HAN is a new heterogeneous graph neural network that uses hierarchical attention, including node-level and semantic-level attentions, to handle this complexity. The semantic-level attention focuses on the significance of each meta-path, while the node-level attention focuses on the importance of a node and its neighbors. The final node embedding is obtained through attention-based aggregations and graph convolutions.

D. Meta-learning

Within the context of the meta-learning paradigm, we look at the few-shot node classification issue. In other words, our goal is to create a classifier that can be adjusted to new classes that weren't present during training, provided a small sample size for each new class. Each node ν_i in the training set D_{train} formally belongs to a class in C1. A given disjoint testing set D_{test}'s nodes are connected to entirely distinct new classes C2. Only a few nodes in D_{test} have labels or classes accessible. Finding a function f with a low misclassification rate that can categorize the remaining unlabeled nodes into one of the classes in C2 is our objective. The job is characterized as a $|C_2|$-way K-shot learning problem, where K is a relatively small integer, if there are K labeled nodes in each class.

III. Data analysis

This section describes the differences we observe between different traffic families from the perspectives of ip address, ports, and Traffic level statistical characteristics (e.g., source bytes, destination bytes). We design the architecture of MetaIoT based on these observations.

According to [15], we know that malicious attackers are resource concentrated. They are unlikely to have a large pool of resources and thus have to reuse the resources they already

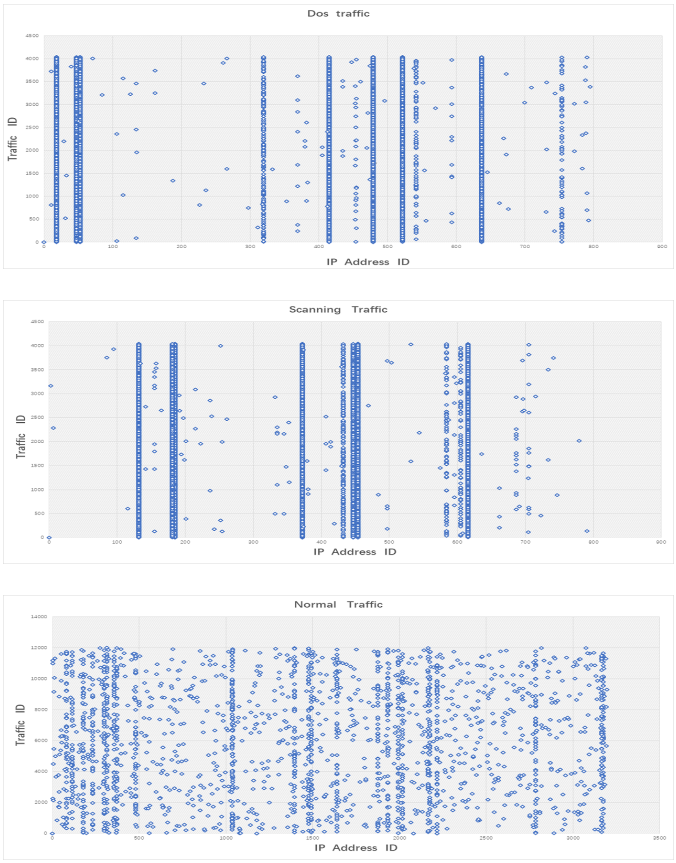
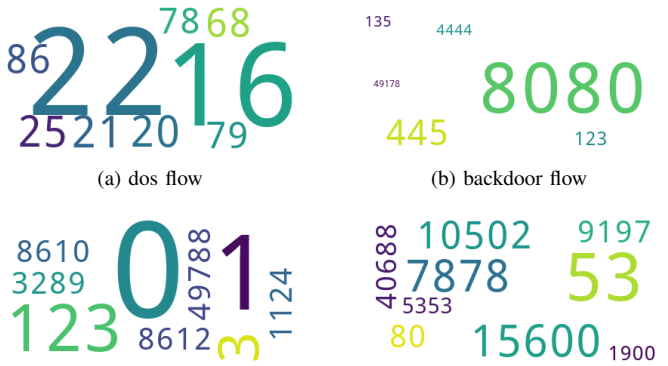


Fig. 1. IP address distribution of traffic

have. This aggregation pattern is especially exhibited for resources such as ip addresses. Figure 1 displays the distributions of IP addresses mapped to dos traffic, scanning traffic and normal traffic. We were able to see that for malicious traffic(dos and scanning traffic), the royal dots display strong signals that some traffic is corresponding to a specific number of IP address. In the meantime, the royal dots disperse more evenly around the IP address space for normal traffic. And different types of traffic show different distribution patterns. We assume that this is the case because attackers are unlikely to possess a large number of IP addresses, hence their distribution shows a relative aggregation trend.

We discovered through the experiments in [16] that various server ports interact with various device kinds. Therefore, we decided to compare the port distribution of different types of traffic in the Ton_IoT dataset to verify whether the port characteristics help us to classify the traffic. Figure 2 displays a word cloud of the source port numbers for three different forms of malicious traffic as well as legitimate traffic. If a port is utilized more frequently for a certain type of flow, it is shown in the corresponding word cloud by a greater font size. It is clear that various traffic types—including dos, backdoor, mitm, and normal—converse with distinct number of server ports. Based on the above findings, we consider traffic, ip address and port as nodes, where the feature vector



(a) dos flow (b) backdoor flow

(c) mitm flow (d) normal flow

Fig. 2. Word-cloud of port numbers for different traffic (the more a port is used, the larger the display font)

of the traffic node consists of time features (duration), service fields and some traffic statistics features (e.g. Number of original/destination packets, Number of original/destination IP bytes).

IV. Methodology

The proposed model MetaIoT has four main components: Data Handler, HIN Constructor, Node Embedding Learner, and Meta Classifier. The data is preprocessed and stored, then an HIN is built to represent the connections in the dataset. To increase efficiency and reduce pollution, some nodes are removed. The GNN is then applied to the HIN to learn node embedding through attention-based aggregations. To handle the challenge of adapting to new classes, the model uses the MAML algorithm for meta-learning. The Meta Classifier is trained to quickly adapt to new tasks using a small number of samples. The performance of MetaIoT is evaluated through meta-testing on new classes.

A. Data Handler

To satisfy our model MetaIoT, we need to transform digital and non-digital features to vector. In regard to string feature like service, connection state and protocol, we use sentence-transformers to compute text embedding. Sentence-transformers framework uses siamese and triplet network structures to generate meaningful text embeddings. These embedding could be used in HAN to compute node similarity. Under the same reason, we use K-bins discretization to partition continuous features into discrete values. For the missing values in the feature, we assign them to a new value to avoid affecting other attribute values.

B. HIN Constructor

MetaIoT neatly models the Ton_IoT dataset as an HIN consists of three different types of nodes, namely traffic, IP addresses, and ports, as illustrated in Figure 3. To represent their relationships, we construct the following adjacency matrices.

R1: To represent relations between traffics and source IP, we build the traffic-from-IP matrix M where element M (i, j) = 1 if traffic i is from IP j , otherwise, M (i, j) = 0 .

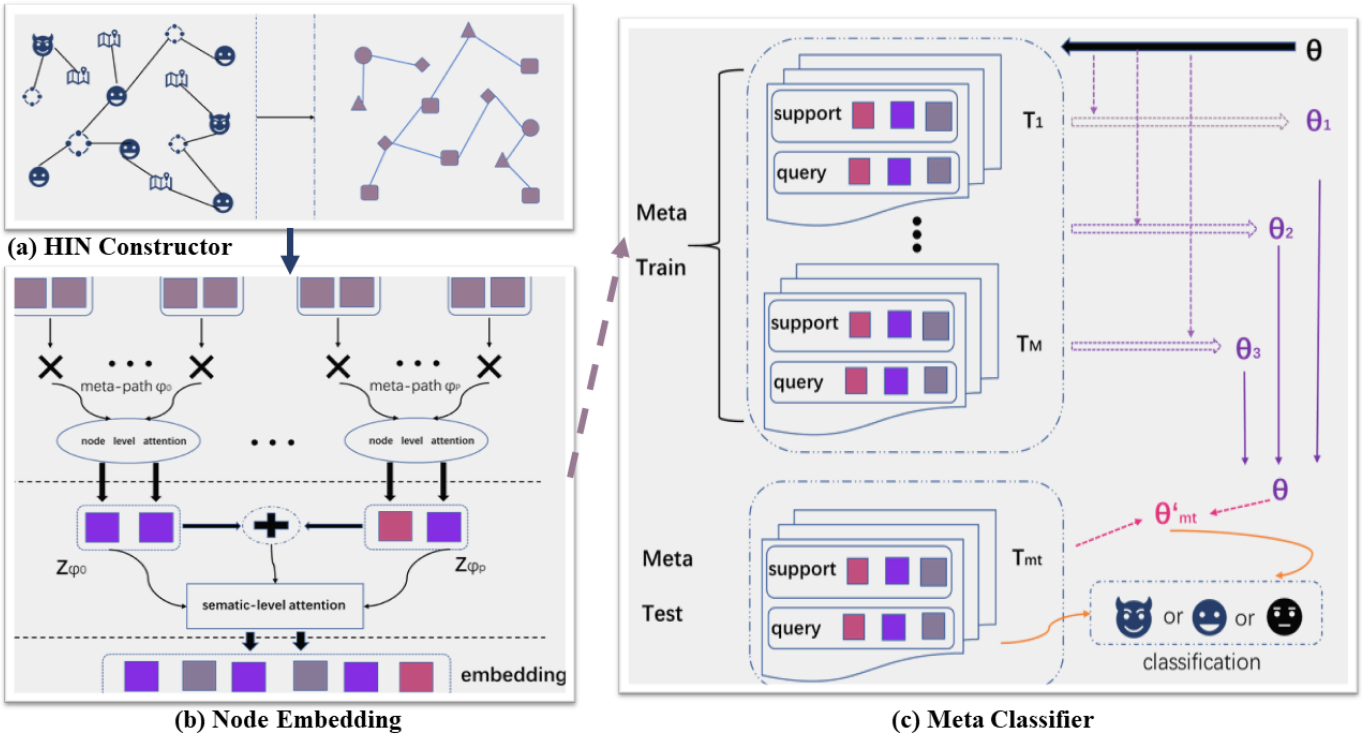


Fig. 3. System architecture of MetaIoT.

R2: To describe relations between traffics and destination IP, we build the traffic-to-IP matrix Q where element Q (i, j) = 1 if traffic i have destination IP j , otherwise, Q (i, j) = 0.

R3: To describe the relation of traffic and source port. We build the traffic-from-Port matrix U where each element U (i,j) ⊂ 0, 1 denotes whether traffic i source from Port j or not.

R4: To describe the relation of traffic and destination port. We build the traffic-to-Port matrix U where each element U (i,j) ⊂ 0, 1 denotes whether traffic i have destination Port j or not.

MetaIoT uses four symmetric meta-paths to identify malicious traffic. P1 focuses on resource aggregation by attackers and P2 uses the port feature to categorize different types of traffic linked to the same port.

C. Node Embeddings Learner

Node-level Attention: We use node-level attention to identify the significance of each traffic node's neighbors based on meta-paths in the heterogeneous graph. We create a type-specific transformation matrix for each node type to project their features into the same space for embedding learning. [17]. Diverse kinds of nodes have different feature spaces as a result of the heterogeneity of nodes. In order to project the features of several node types into the same feature space, we create the type-specific transformation matrix A_{ϕ_i} for each kinds of nodes. The following diagram illustrates the projection process:

$$s_i^{'} = A_{\phi_i} \cdot s_i \tag{1}$$

We transform node i's original features $(s^{'})$ into projected features (s_i) using a type-specific transformation matrix. Then, we use self-attention to weigh the significance of each node type. The node-level attention r_{ij}^{ϕ}, or the significance of node j to node i in a meta-path-connected node pair (i,j), is calculated as follows:

$$r_{ij}^{\phi} = Att(s_i^{'}, s_j^{'}; \phi). \tag{2}$$

The deep neural network Att is used here to execute node-level attention. Given a meta-path, the preceding Eq.(2) demonstrates that the weight of a meta-path-based node pair (i,j) relies on its characteristics. Then we calculate the r_{ij}^{ϕ} for j $\in \mathcal{N}$, where $\mathcal{N}$ stands for the neighbors of node i based on the meta-path (including itself). The weight coefficient ξ_{ij}^{ϕ} is obtained using the softmax function after normalizing the relevance between meta-path based node pairs:

$$\xi_{ij}^{\Phi} = \mathrm{softmax}_j\left(r_{ij}^{\Phi}\right) = \frac{\exp\left(\sigma\left(\mathbf{v}_{\Phi}^{\mathrm{T}} \cdot \left[\mathbf{s}_i' \| \mathbf{s}_j'\right]\right)\right)}{\sum_{k \in \mathcal{N}_i^{\Phi}} \exp\left(\sigma\left(\mathbf{v}_{\Phi}^{\mathrm{T}} \cdot \left[\mathbf{s}_i' \| \mathbf{s}_k'\right]\right)\right)} \tag{3}$$

where σ denotes the activation function, $\|$ denotes the concatenate operation and $\mathbf{v}_{\phi}$ is the node-level attention vector for meta-path ϕ. The weight coefficient ξ_{ij}^{ϕ} of (i,j) is dependent on their characteristics, as can be shown from Eq. (3).

The neighbor's projected features may then be combined with the associated coefficients to aggregate the meta-path based embedding of node i as shown below:

$$\mathbf{z}_i^{\Phi} = \sigma\left(\sum_{j \in \mathcal{N}_i^{\Phi}} \xi_{ij}^{\Phi} \cdot \mathbf{s}_j'\right) \tag{4}$$

where z_i^{ϕ} is the learned embedding of node i for the meta-path ϕ.

We extend node-level attention to multihead attention to address the aforementioned problem and increase the stability of the training process. To create the semantic-specific embedding, we concatenate the learnt embeddings after repeating the node-level attention N times.

$$\mathbf{z}_i^{\Phi} = \|_{n=1}^{N} \sigma\left(\sum_{j \in \mathcal{N}_i^{\Phi}} \xi_{ij}^{\Phi} \cdot \mathbf{s}_j'\right) \tag{5}$$

Given the meta-path set $\{\phi_1, ..., \phi_P\}$, we may acquire p sets of semantic-specific node embeddings after feeding node characteristics into node-level attention, labeled as $\{\mathrm{Z}_{\Phi_1}, \ldots, \mathrm{Z}_{\Phi_P}\}$.

Semantic-level Attention: Semantic-specific node embeddings can only represent a node from one aspect. We need to combine several semantics, which meta-paths can reveal, to develop a more thorough node embeddings [16]. Using P sets of semantically distinct node embeddings discovered by node-level attention as input, the learned weights of each meta-path $\{\varepsilon_{\Phi_1}, \ldots, \varepsilon_{\Phi_P}\}$ can be shown as follows:

$$\left(\varepsilon_{\Phi_1}, \ldots, \varepsilon_{\Phi_P}\right) = \mathrm{Att}\ \left(\mathbf{Z}_{\Phi_1}, \ldots, \mathbf{Z}_{\Phi_P}\right) \tag{6}$$

The deep neural network named Att is used here to carry out semantic-level attention.

We evaluate the importance of each meta-path by nonlinearly transforming the semantic-specific embedding and comparing it with a semantic-level attention vector. The meta-paths are then weighted based on their relevance when averaging all the semantic-specific node embeddings. The importance of each meta-path, denoted as τ_{ϕ_i} , is shown as follows:

$$\tau_{\Phi_p} = \frac{1}{|\mathcal{V}|} \sum_{i \in \mathcal{V}} \mathbf{q}^{\mathrm{T}} \cdot \tanh\left(\mathbf{W} \cdot \mathrm{z}_i^{\Phi_p} + \mathbf{b}\right) \tag{7}$$

where $\mathbf{q}$ is the semantic level attention vector, $\mathbf{b}$ is the bias vector, and $\mathbf{W}$ is the weight matrix. We use the softmax function to normalize each meta-path after determining its significance. By using the softmax function to normalize the above significance of all meta-paths, it is possible to determine the weight of meta-path ϕ_i denoted as β_{Φ_p}.

$$\varepsilon_{\Phi_p} = \frac{\exp\left(\tau_{\Phi_p}\right)}{\sum_{p=1}^{P} \exp\left(\tau_{\Phi_p}\right)} \tag{8}$$

This might be interpreted as the meta-path Φ_p's contribution to a particular task. Naturally, the larger ε_{Φ_p}, the more significant the meta-path Φ_p is. Keep in mind that the weights of metapath Φ_p might vary depending on the job. We can combine these semantically specialized embeddings using the learned weights as coefficients to get the final embedding Z, which looks like this.

$$\mathbf{Z} = \sum_{p=1}^{P} \varepsilon_{\Phi_p} \cdot \mathbf{Z}_{\Phi_p} \tag{9}$$

All semantic-specific embeddings are combined to form the final embedding. The finished embedding may then be applied to certain tasks and alternative loss functions can be designed.

D. Meta Classifier

After the Node Embeddings Learner, the nodes on the graph already learn the relationship knowledge between nodes through HAN. We now demonstrate how to employ these embeddings to resolve the few-shot IoT traffic classification learning issue (also known as the $\mathcal{T}_{mt}$ meta-testing task) by training on previously sampled similar tasks (i.e., meta-training tasks).

The training and testing sets that correspond to each task are referred to as the support set and query set, respectively. MetaIoT is anticipated to learn how to swiftly adjust to a new type of traffic after training on a significant amount of meta-training tasks (as prior knowledge). The gradient updates in our method's training phase use MAML [18]. By meta-testing the new job, or fine-tuning MetaIoT on a few samples from the support set of $\mathcal{T}_{mt}$ and assessing it on the query set of $\mathcal{T}_{mt}$, the performance of MetaIoT is evaluated.

We denote our MetaIoT model as g_θ with parameters θ, and the training set as $\mathcal{M}_{\text{train}} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_i, y_i), \ldots, (\boldsymbol{x}_N, y_N)\}$ where yi∈B1 and N is the number of nodes in the training set. Depending on the dataset we use, y has a total of 10 classes. From $\mathcal{M}_{train}$, we will divide M meta-training tasks as follows: $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_M\}$.

In the support set, we have $\mathcal{S}_i = \{v_{i1}, v_{i2}, \ldots, v_{is}\} = \{(\boldsymbol{x}_{i1}, y_{i1}), (\boldsymbol{x}_{i2}, y_{i2}), \ldots, (\boldsymbol{x}_{is}, y_{is})\}$ where s=$|\mathcal{S}_i|$; x_{is} is the input vector of node v_{is}(represented as a traffic node) with label y_{is}.

Task Sampling: By selecting tasks from $\mathcal{M}_{train}$, we create M tasks $\mathcal{T}$. To mimic few shot node classification, we sample $|B2|$ classes from B1 and then randomly sample K nodes for each class. That is, there are only a few K flows in each class of traffic as a support set. Below are the main steps.

- $B \leftarrow CHOOSE\ \ |B_2|\ \ classes\ \ FROM\ \ B1$;
- $S_i \leftarrow CHOOSE\ \ K \times |B_2|\ \ nodes\ \ FROM\ \ \mathcal{M}_B$;
- $Q_i \leftarrow CHOOSE\ \ L\ \ nodes\ \ FROM\ \ \mathcal{M}_C - \mathcal{S}_i$;
- $\mathcal{T}_i = \mathcal{S}_i + Q_i$;
- Repeat step (1) - (4) for M times;

Therefore, we begin by selecting $|B2|$ classes at random from B1, also known as B. Once we have $\mathcal{M}_C$, which is a subset of training set M_{train} with elements (**x***i* , yi), where yi is one of the classes in B. To create the support set $\mathbf{S}_i$, we then randomly select K × $|B2|$ nodes from $\mathcal{M}_C$, where K is the number of nodes in each class of $\mathbf{S}_i$ (i.e., the number of shots). Finally, we create the query set Q_i from a random sample of L nodes taken from the $\mathcal{M}_C$'s remaining nodes.

This query set is then utilized to create the meta-training task $\mathcal{T}_i = \mathbf{S}_i + \mathbf{Q}_i$. The preceding processes repeated M times result in M meta-training tasks.

Meta-training: We aim to obtain a good initialization of the Meta Classifier for related tasks by initializing its parameters to perform well after a few gradient descent updates on a new few-shot learning task. We next pass the support set $\mathbf{S}_i$ to the Meta Classifier when learning a task $\mathcal{T}_i$, and then we calculate the cross-entropy loss:

$$\begin{aligned}\mathcal{L}_{\mathcal{T}_i}(\boldsymbol{\theta}) &= (y_{is}-1)\log\left(1-g_{\boldsymbol{\theta}}(\boldsymbol{x}_{is})\right) \\ &- \sum_{\boldsymbol{x}_{is},y_{is}} y_{is}\log g_{\boldsymbol{\theta}}(\boldsymbol{x}_{is})\end{aligned} \tag{10}$$

Then, using a straightforward gradient descent with one or more steps in task $\mathcal{T}_i$, we update the parameters. For the sake of conciseness, we will just briefly discuss one gradient update in the next portions of this section, with a warning that executing multiple gradient updates is a simple extension:

$$\theta_i' = \theta - lr_1 \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(g_\theta) \tag{11}$$

where the model parameters are trained to enhance the performance of $g_\theta^{'}$ across meta-training tasks, and lr_1 is the task-learning rate. The meta-objective is more explicitly as follows:

$$\boldsymbol{\theta} = \min_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta - lr_1 \nabla_\theta \mathcal{L}_{\mathcal{T}_i}(f_\theta)}\right) \tag{12}$$

where the distribution of meta-training tasks is denoted by $p(\mathcal{T})$. The goal is calculated using the model parameters θ. This is due to the fact that we require effective initialization parameters θ for all related few-shot node classification tasks rather than updated settings $\theta_i^{'}$ that are effective on only one job $\mathcal{T}_i$. In essence, our model seeks to maximize node classification performance on a new job(carries only a few samples) after a minimal number of gradient descent updates by optimizing the model parameters. The model parameters are updated in the following equation, where stochastic gradient descent (SGD) is used to conduct the meta-optimization across jobs and lr_2 is the meta-learning rate..

$$\theta \leftarrow \theta - lr_2 \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right) \tag{13}$$

Meta-testing: To do meta-testing, all required is to feed the Meta classifier with the nodes from the support set of the new few-shot learning task (i.e., $\mathcal{T}_{mt}$, some categories that were not learned during previous training) and update the parameters of $\theta_{mt}^{'}$ using one or a few gradient descent steps using Eq (11). Therefore, using $\mathcal{T}_{mt}$'s query set, it is simple to assess MetaIoT's performance and thus solve the problem of detecting IoT malicious traffic in small sample scenarios.

Algorithm 1 Training Procedure of MetaIoT.

Input: Graph adjaccent matrix: A; Node features: s; Meta-path set: ϕ; Initial parameters: δ; Task-learning rate: lr_1; Meta-learning rate: lr_2; distribution $p(\mathcal{T})$ over $\mathcal{M}_{train}$; meta-testing tasks: $\mathcal{T}_{mt}$
Output: Labels of nodes in query set of $\mathcal{T}_{mt}$.
foreach $\phi_i \in \{\ \phi_0,\ \phi_1,\ ...,\ \phi_P\}$ **do**
  foreach *node i in HIN* **do**
    Find neighbors $\mathcal{N}_i^\Phi$ based meta-path ϕ_i
    foreach $j \in \mathcal{N}_i^\Phi$ **do**
      Calculate the weight coeffcient ξ_{ij}^ϕ
    end
    Concatenate the node-level attention embedding $\mathbf{z}_i^\Phi = \|_{n=1}^N \sigma\left(\sum_{j\in\mathcal{N}_i^\Phi} \xi_{ij}^\Phi \cdot \mathbf{s}_j'\right)$
  end
  Calculate the weight of meta-path ε_{Φ_i};
  Fuse semantic-level attention embedding $\mathbf{Z} = \sum_{p=1}^P \varepsilon_{\Phi_p} \cdot \mathbf{Z}_{\Phi_p}$
end
Initialize θ randomly;
while *not converged* **do**
  Sample batch of meta-training tasks $\mathcal{T}_i \sim p(\mathcal{T})$;
  foreach $\mathcal{T}_i$ **do**
    Calculate $\mathcal{L}_{\mathcal{T}i}(f_\theta)$ using $\mathcal{S}_i$;
    Update adapted parameters $\theta_i^{'}$;
    Calculate $\mathcal{L}_{\mathcal{T}i}(f_{\theta_i'})$ using $\mathcal{Q}_i$;
  end
  Update $\theta \leftarrow \theta - lr_2 \nabla_\theta \sum_{\mathcal{T}_i \sim p(\mathcal{T})} \mathcal{L}_{\mathcal{T}_i}\left(f_{\theta_i'}\right)$
end
Update parameters $\theta_{m,t}'$ using $\mathcal{S}$ in $\mathcal{T}_{mt}$;
Predict labels of $\mathbf{Z}$ in $\mathcal{Q}$ of $\mathcal{T}_{mt}$ using model $f_{\theta_{mt}'}$.

V. Experiments

As previously mentioned, we tested the effectiveness of our suggested MetaIoT model on several threat categories using the Ton_IoT dataset. The following two questions will be addressed in this section: (a) How well did the MetaIoT model perform in the circumstance of a few shot sample? (b) How did the model perform when enough sample cases were used? We create original sample situations and tiny sample scenarios for the two difficulties mentioned above in order to conduct evaluation studies.

A. Data Preprocessing

For Ton_IoT dataset, we first use the definition of a five tuple (src_ip, dst_ip, src_port, dst_port, protocol) to define a traffic flow, and we perform the drop duplication operation for repeated traffic. In most realistic settings, the number of malicious traffic samples is usually much smaller than that of normal traffic. Table I displays the statistics of the datasets after pre-processing. In this case, malicious traffic samples account for only about 3% of all traffic. Then, according to the description in section IV, we generated an IOT heterogeneous graph.

B. Few-shot Sample Learning Ability

For the Ton_IoT datasets, we choose two classes at random to serve as meta-testing classes (i.e., the associated nodes forming $\mathcal{M}_{test}$ with $|B2|= 2$), and the remaining nodes (i.e., $\mathcal{M}_{train}$) are utilized to build meta-training tasks using the technique outlined in section 2. 2. There are only K samples for each class in the support set of the meta-training and meta-testing tasks (K = 3, K = 5, or K = 10) for all datasets. We found that the node selection has an impact on each method's performance when the support set is quite limited. As a result, we assess each model using data from the same batch and present the average accuracy. To ensure a balanced division of the dataset, we used stratified sampling to create the training, validation, and testing subsets. The subsets were created such that they reflected the class proportions of the original dataset, with 80% of the samples used for training, 10% used for validation, and 10% used for testing.

TABLE I
TON_IOT DATASET FOR FEW-SHOT CLASSIFICATION

	Traffic Type						
	Normal	Scanning	Injection	DDoS	Password	XSS	DoS
Number	26140	120	130	122	136	134	89
	MITM	Backdoor	Ransomware				
	110	77	91				
	Descriptive Statistics and Partition of Ton_IoT Dataset						
	Nodes	Edges	$\|C1\|$	$\|C2\|$			
	27140	2M	5	2			

Baselines: For Ton_IoT, we compare MetaIoT against DeepWalk [19], GraphSAGE [20], SGC [21], GCN [22], HAN [8], Meta-GCN, Meta-SGC. The reason we chose these models is because they are representative models in graph neural network models, among which GraphSAGE and DeepWalk could not be compared with the combination of meta-learning due to memory limitations, but for the rest of the models, we compared them under the conditions of ensuring a unified dataset and optimal parameters. Notably, we only alter the dataset partition to fulfill the meta-learning paradigm's few-shot learning requirements; all other model variables remain unchanged from their initial implementation.

Implementation and Parameter Setups: In order to enhance the convergence of our model, we modified the batch size in Algorithm 1 to 12. The learning rates, denoted as $lr1$ and $lr2$, were set to 0.1 and 0.001, respectively. For HAN, we set the learning rate to 0.005, the regularization parameter to 0.001, the dimension of the semantic-level attention vector to 128, the number of attention heads to 8, and the attention dropout to 0.6. Additionally, we employed early stopping with a patience of 100, meaning that if the validation loss does not decrease for 100 consecutive epochs, the training process is ceased. The parameters of GCN and SGC were optimized utilizing the validation set. To ensure fairness, the same training, validation, and test sets were employed for all models. For random walk-based methods, such as DeepWalk, we set the window size to 5, walk length to 100, walks per node to 40, and number of negative samples to 5. Lastly, to guarantee a fair comparison, the embedding dimension was set to 64 for all algorithms. The remaining model settings were kept the same as the recommendations in their original publications.

Experimental Result: The top performances are bolded in Table II's results of the performance comparison between MetaIoT and baselines. We can see from the table that our suggested model performs the best overall across all models. Even when compared to GCN and SGC, GraphSAGE has not produced results that are competitive. This finding suggests that earlier inductive graph learning methods do not transfer well to new classes, while having demonstrated promising results when dealing with new nodes (as described in the original study [22]). Because heterographs gain more relation information than homogeneous graphs, they outperform SGC and GCN in terms of performance. In general, heterograph models—like ours—perform much better on few-shot learning situations than homogeneous graph-based models, such as GCN and SGC. A similar conclusion can also be seen in Figure 4.

Hyper-parameter Sensitivity: We investigate MetaIOT sensitivity to its two hyperparameters: the dimension of the final embedding and the dimension of the semantic-level attention vector under different k-shot scenarios. Figure 5 depicts the F1 score of the aforementioned metrics, respectively.

As can be seen from the results, Meta-IOT's performance improves as the dimension of the final embedding increases. However, when the dimension of the final embedding exceeds 64, it only brings about a minimal improvement in performance. The reason is that HAN needs a suitable dimension to encode the semantics information and larger dimension may introduce additional redundancies. Hence, it is recommended to set the embedding dimension to 64. Regarding the semantic-level attention vector, Meta-IOT's performance also gradually improves with an increase in dimension and number of samples in small sample categories. However, the performance begins to decline as the vector continues to increase, which could be due to overfitting.

C. Learning Ability from Ordinary Sample Scenarios

To answer question (b), We directly put The Ton_IoT overall data set into the HIN constructor and learned node embeddings. In order to maintain consistency with existing methods for data splitting, the dataset was divided into 60% for training, 20% for validation, and 20% for testing. Finally, we use these embedding and cross entropy loss functions to classify nodes. Table III details the performance of the suggested heterogeneous detection model when applied to the Ton_IoT dataset. The evaluation's findings show that when compared to 4 other models, the model produces greater results. The suggested model outperformed all techniques considered, with an f-score of 98.60%. Figure 6 presents the full confusion matrix of our classification results. The matrix showcases the distribution of the true classifications versus the predicted ones.

Based on the above experiments, we verified MetaIoT effect for few shot sample scenes and sufficient sample scenes.

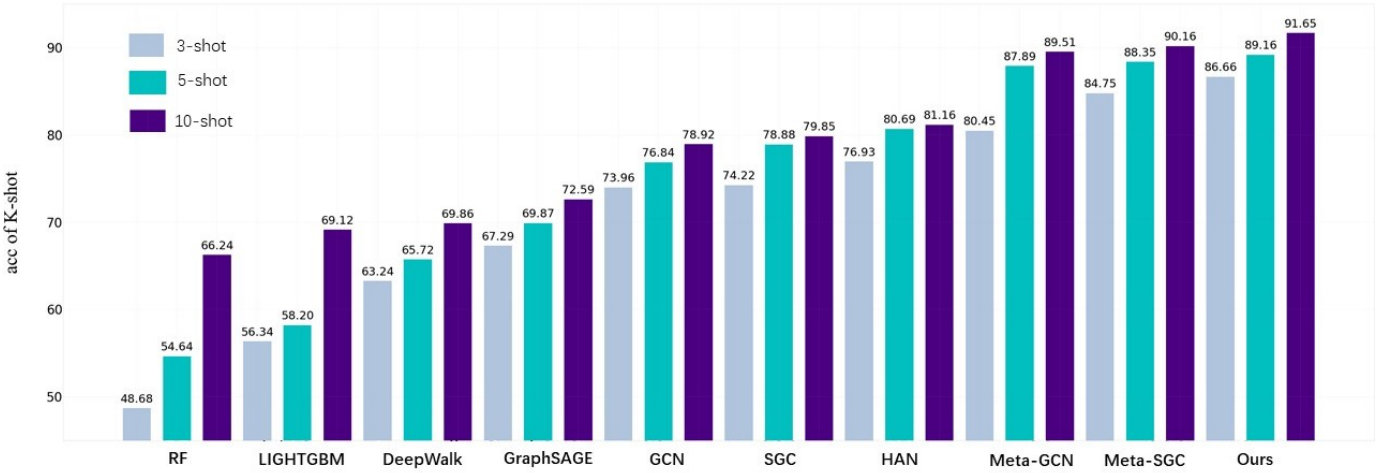
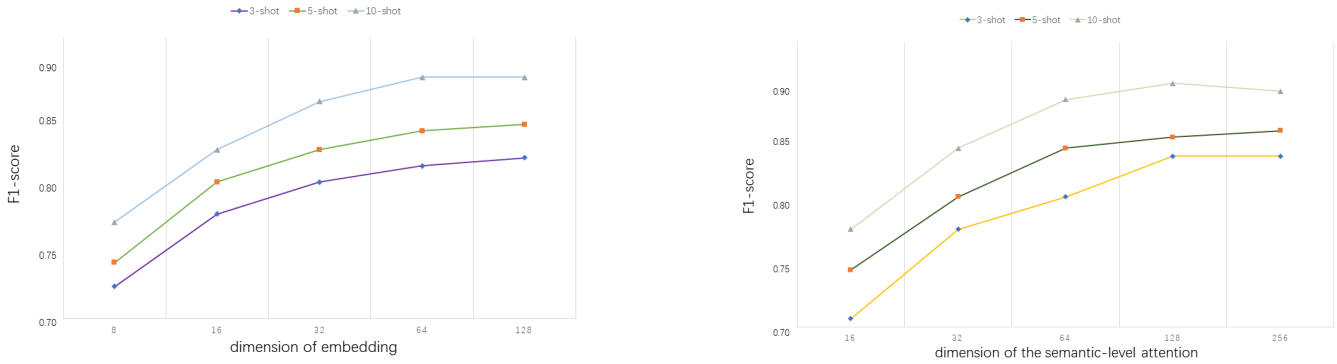


Fig. 4. Performance comparisons of all methods with different support data sizes (3-shot, 5-shot or 10-shot).



(a) Dimension of the final embedding

(b) Dimension of the semantic-level attention vector

Fig. 5. F1-score of Different Final Embedding Dimensions and Different Dimensions of Semantic Attention on Different k-shot Settings

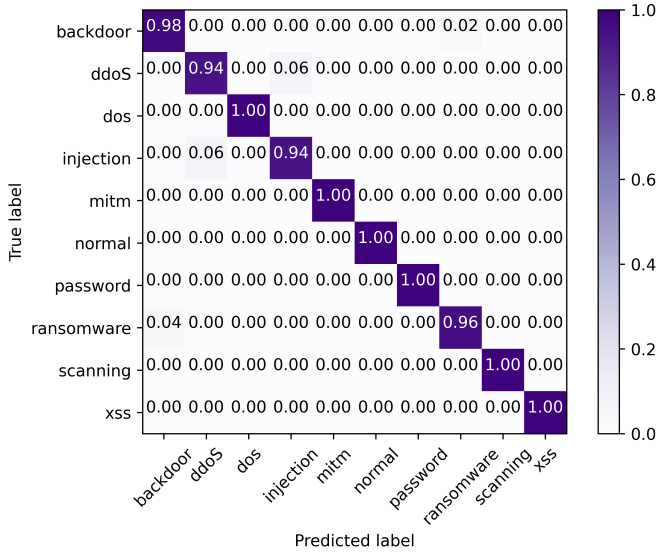


Fig. 6. Performance comparison with existing methods

TABLE II
PERFORMANCE COMPARISON WITH EXISTING METHODS USING THE OVERALL DATA SET.

Model	A(%)	P(%)	R(%)	F(%)
Nguyen et al. [11]	99.09	**99.91**	96.34	97.77
Schneble et al. [12]	98.17	98.79	96.45	97.57
Yingying et al. [13]	94.28	88.18	84.32	86.20
Booij, T. M. [14]	98.07	-	-	97.26
ours	**99.20**	98.86	**98.36**	**98.60**

In all cases, MetaIoT significantly outperforms all baseline methods, demonstrating the strongest capability of our model for malicious IoT traffic detection.

VI. Limitation and Future Work

This work achieves IoT network intrusion detection from limited supervised information, in order to wean IDS from dependency on large-scale labeled datasets. However, detection capacity of MetaIoT still derives from specific malicious samples. Ideally, we hope MetaIoT can detect unseen attacks even if security experts know nothing about them. Besides,

in terms of detection accuracy and recall comparing with supervised learning on large-scale datasets, MetaIoT still has potential for making further progress. We think the key issues for advancement are more representative features of network flow. How to design a set of comprehensive and representative features for describing network behaviors accurately remains a problem for security researchers. Lastly, the selection of meta-paths affects traffic embedding, and we aim to develop a method to learn traffic embedding directly from heterogeneous graphs without relying on expert experience. Addressing these challenges will be our focus for future work.

VII. Conclusion

In this paper, to solve the problem of malicious IoT traffic detection with limited labeled data constraints, we use the standard IoT dataset (Ton_IoT) and develop a novel system called MetaIoT. To be more precise, we first discussed feature selection, created an HIN to extract the dataset's complex semantics structural relationships and node content, then used a hetero GNN to train the node embeddings. Afterwards, we design a meta-learning algorithm to optimize the model, so that the model can complete target detection in few shot sample scenarios. The extensive results demonstrate the effectiveness of our model by comparison with many baseline models.

Acknowledgment

This work is supported by the Strategic Priority Research Program of the Chinese Academy of Sciences with No. XDC02030000, The National Key Research and Development Program of China No. 2021YFB3101403 and National Key R&D Program 2021(Grant No. 2021YFB3101001).

References

[1] C. Anna and D. Asher, "Risks in iot supply chain," 2020, https://unit42.paloalTonetworks.com/IoT-supply-chain, Last accessed on April 25, 2022.

[2] D. Jun, L. Derick, and D. Aveek, "Windows xp, server 2003 source code leak leaves iot, ot devices vulnerable," 2020, https://unit42.paloaltonetworks.com/windows-xp-server-2003-source-code-leak/, Last accessed on April 25, 2022.

[3] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "Iot network traffic classification using machine learning algorithms: an experimental analysis," *IEEE Internet of Things Journal*, vol. 9, no. 2, pp. 989–1008, 2021.

[4] B. Charyyev and M. H. Gunes, "Iot traffic flow identification using locality sensitive hashes," in *ICC 2020-2020 IEEE International Conference on Communications (ICC)*. IEEE, 2020, pp. 1–6.

[5] M. Al-Hawawreh, N. Moustafa, S. Garg, and M. S. Hossain, "Deep learning-enabled threat intelligence scheme in the internet of things networks," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 4, pp. 2968–2981, 2020.

[6] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[7] I. A. Khan, N. Moustafa, I. Razzak, M. Tanveer, D. Pi, Y. Pan, and B. S. Ali, "Xsru-iomt: Explainable simple recurrent units for threat detection in internet of medical things networks," *Future generation computer systems*, vol. 127, pp. 181–193, 2022.

[8] C. Shi, Y. Li, J. Zhang, Y. Sun, and S. Y. Philip, "A survey of heterogeneous information network analysis," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2016.

[9] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu, "Pathsim: Meta path-based top-k similarity search in heterogeneous information networks," *Proceedings of the VLDB Endowment*, vol. 4, no. 11, pp. 992–1003, 2011.

[10] X. Wang, H. Ji, C. Shi, B. Wang, Y. Ye, P. Cui, and P. S. Yu, "Heterogeneous graph attention network," in *The world wide web conference*, 2019, pp. 2022–2032.

[11] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for iot," in *2019 IEEE 39th International conference on distributed computing systems (ICDCS)*. IEEE, 2019, pp. 756–767.

[12] W. Schneble and G. Thamilarasu, "Attack detection using federated learning in medical cyber-physical systems," in *28th International conference on computer communications and networks (icccn)*, 2019, pp. 1–8.

[13] Y. Xu, Z. Liu, Y. Li, H. Hou, Y. Cao, Y. Zhao, W. Guo, and L. Cui, "Feature data processing: Making medical data fit deep neural networks," *Future Generation Computer Systems*, vol. 109, pp. 149–157, 2020.

[14] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. den Hartog, "Ton_iot: The role of heterogeneity and the need for standardization of features and attack types in iot network intrusion data sets," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2021.

[15] X. Sun, Z. Wang, J. Yang, and X. Liu, "Deepdom: Malicious domain detection with scalable and heterogeneous graph convolutional networks," *Computers & Security*, vol. 99, p. 102057, 2020.

[16] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying iot devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.

[17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[19] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2014, pp. 701–710.

[20] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[21] F. Wu, A. Souza, T. Zhang, C. Fifty, T. Yu, and K. Weinberger, "Simplifying graph convolutional networks," in *International conference on machine learning*. PMLR, 2019, pp. 6861–6871.

[22] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.