A Graph Neural Network Method for Distributed Anomaly Detection in IoT

Aikaterini Protogerou[1] · Stavros Papadopoulos[2] · Anastasios Drosou[3] · Dimitrios Tzovaras[4] · Ioannis Refanidis[5]

Abstract Recent IoT proliferation has undeniably affected the way organizational activities and business procedures take place within several IoT domains such as smart manufacturing, food supply chain, intelligent transportation systems, medical care infrastructures etc. The number of the interconnected edge devices has dramatically increased, creating a huge volume of transferred data susceptible to leakage, modification or disruption, ultimately affecting the security level, robustness and QoS of the attacked IoT ecosystem. In an attempt to prevent or mitigate network abnormalities while accommodating the cohesiveness among the involved entities, modeling their interrelations and incorporating their structural, content and temporal attributes, graph-based anomaly detection solutions have been repeatedly adopted. In this article we propose, a multi-agent system, with each agent implementing a Graph Neural Network, in order to exploit the collaborative and cooperative nature of intelligent agents for anomaly detection. To this end, against the propagating nature of cyber-attacks such as the Distributed Denial-of-Service (DDoS), we propose a distributed detection scheme, which aims to monitor efficiently

[1] University of Macedonia, Greece
156, Egnatia str.
E-mail: k.protogerou@uom.edu.gr (corresonding author)

[2] Information Technologies Institute - CERTH, Greece
6th km Harilaou - Thermis
E-mail: spap@iti.gr

[3] Information Technologies Institute - CERTH, Greece
6th km Harilaou - Thermis
E-mail: drosou@iti.gr

[4] Information Technologies Institute - CERTH, Greece
6th km Harilaou - Thermis
E-mail: Dimitrios.Tzovaras@iti.gr

[5] University of Macedonia, Greece
156, Egnatia str.
E-mail: yrefanid@uom.edu.gr

the entire network infrastructure. To fulfill this task, we consider employing monitors on active network nodes such as IoT devices, SDN forwarders, Fog Nodes, achieving localization of anomaly detection, distribution of allocated resources such as the bandwidth and power consumption and higher accuracy results. In order to facilitate the training, testing and evaluation activities of the Graph Neural Network algorithm, we create simulated datasets of network flows of various normal and abnormal distributions, out of which we extract essential structural and content features to be passed to neighbouring agents.

Keywords IoT cybersecurity · graph inherent anomaly detection framework · graph neural networks · DDoS attack detection · decentralized detection · synergistic detection · multi-agent detection

1 Introduction

Smart IoT devices including sensors, processors, smart cameras, wearables, actuators, and smart vehicles have been rapidly spreading across several IoT application domains, such as smart home, medical and healthcare, transportation, building, industry 4.0 manufacturing, as well as agriculture, and food supply chain environments. Along with the evolving of the IoT ecosystems, adversarial activities have been expanding and becoming more severe, targeting the system's robustness, thus affecting latency sensitive applications. Anomalies within the network context are seen as the utilization of resources in an unusual, not abiding to the norm manner. Evidently, the urgent need to prevent or mitigate occurring anomalies and sustain the global stability, arises in conjunction with the need to enhance already existing techniques so as to address the challenges posed by the high complexity of the anomalies and lack of labeled data.

Given the undergoing ascent of Artificial Intelligence (AI), Machine Learning (ML) algorithms have gained wide acceptance and have been extensively used to detect irregular network traffic patterns. Attempting to control the consequences of upcoming/undergoing cyber-attacks and equipment's physical deficiency, Neural Networks (NN) have been employed to learn the patterns of the transmitted data, aiming to classify abnormal cases, thus ensuring great levels of trustworthiness. With the adoption of NN detection strategies, security breaches are depicted as non-typical data patterns which ultimately reveal the existence of a malware, botnet or an infrastructure failure when compared to the normal state.

Along with the incorporation of the latter techniques, graph-based anomaly detection studies have come into the limelight. In network-based anomalies, graphs constitute an effective way to represent inter-connectivities among the network's nodes and their inter-relations. Structural, behavioral and temporal data are associated to the nodes and edges of the network, creating patterns of behavior which define normal or deviating behavior in an attempt to protect the network's seamless operation and the continuous provision of time-sensitive services as in the cases of critical IoT infrastructures.

This article, addresses the complexity of the underlying IoT network infrastructure, by employing a Graph Neural Network (GNN) model. We propose an anomaly detection solution with high detection accuracy, which takes into account the structural features and relations among neighboring nodes and their attached edges, within the IoT ecosystem. Furthermore, we address the resource constraints posed by the incorporation of the IoT devices, thus proposing an efficient detection scheme of low bandwidth and power-consumption requirements. Against the challenge introduced by the distributed nature of cyber-attacks, we deploy a distributed multi-agent system (MAS), thus addressing the problem of a compromised node being unaware of its infection, giving erroneous signs regarding its own status or going suddenly offline. The distributed MAS architecture will employ anomaly detection on active nodes of the IoT network, such as the Fog Nodes (FN) or SDN forwarders, engaging a GNN to accommodate the physical network's representation. Information exchange among neighbouring agents will enable them to identify propagating distributed attacks and protect the network in a synergistic manner. The absence of a central Intrusion Detection System (IDS) to monitor and analyse all traffic transferred through the network nodes, eliminates the additional overhead in terms of bandwidth and power consumption.

The rest of the article is organized as follows: Sec.2 presents background details concerning various architectures of Neural Networks, which have been used for Anomaly Detection. Sec.3 describes the proposed methodology of implementing the GNN-based anomaly detection method, the feature extraction process and presents the dataset we used in the experiments. Sec. 4 presents the evaluation results of the proposed method compared to traditional ML techniques and two state-of-the-art algorithms along with the generation of Infiltration and Worm attack propagation datasets. Sec. 5 elaborates on the application domains potentially adopting the proposed approach and limitations posed by hardware requirements. Finally, Sec.6 concludes the article and sets future plans and up-to-date considerations.

2 Related work

Following the prominence of NN and Deep Learning methodologies, a lot of research has been conducted over the past few years towards, compbining and optimizing existing algorithms in order to discover the best detection approach. Apart from the latter, graph-based solutions have been proposed to identify anomalies on the nodes and edges along with clustering techniques. Probabilistic methods taking into account historical data by utilizing Bayesian networks have been considered, as well. This sections reviews some of the most prominent recent anomaly detection techniques.

2.1 Hybrid NN-based anomaly detection approaches

In [28] J. Kim. et al. implemented a hybrid LSTM-RNN detection model in which they exchanged the RNN classifier by using an LSTM cell for the recurrent hidden layer. Another hybrid model was presented in [57], where a combination of CNN and LSTM model was proposed for learning Hierarchical Spatial-Temporal Features (HAST-IDS). Validation of the proposed model proved the influence of the network packet size, the network flow size and the number of network packets on the chosen metrics of Accuracy, DR and FAR.

M. Cheng et al. [10] proposed a Deep Learning (DL) model to analyze time series of traffic data upon the Border Gateway Protocol (BGP). Past feature values were utilized to perform current feature values classification, proposing an anomaly detection method combining Multi-Scale LSTM (MS-LSTM) and pre-processing steps.

S. M. Erfani et al. in [13] proposed a hybrid architecture of a Support Vector Machine (SVM) integrated with a Deep Belief Network (DBN) method to produce a model to alleviate the volume of high-dimensional data by extracting low dimensional features. SVM separated normal from malicious data, whereas the DBN was selected as the dimensionality reduction and feature extraction algorithm.

A self-adaptive anomaly detection method was proposed in [36] with respect to 5G networks. The authors proposed a DBN in conjunction with a Stacked Auto-Encoders (SAE) hybrid model, to detect local abnormal traffic during a flexible time window and an LSTM Recurrent model to run afterwards on the component responsible for refining detection results. Another self-taught-learning (STL) model is proposed in [25] for network anomaly detection. In this study, they used a sparse auto-encoder network with backpropagation. A similar approach is described in [1], where the authors used hybrid model of sparse auto-encoders trained with the unlabeled dataset, followed by an SVM to classify the examples.

A DL method for anomaly detection in a IEEE 802.11 network scenario was the case in [53]. The dataset utilized by the authors contained real data traffic of a lab emulated SOHO infrastructure. Flooding, injection and impersonation attacks were classified by the developed Stack Auto-encoders model.

2.2 Deep Learning approaches for anomaly detection in IoT network infrastructures

In [60] F. Y. Yavuz et al. simulated an IoT network testbed, utilizing the Contiki operating system. They detected Routing Attacks, employing a DL method and extracting the data needed out of the sensors raw data with the combination of Random Decision Trees and Pearson coefficient correlation.

Intrusion detection for in-vehicle security using a Deep Neural Network method is described in [26]. The authors proposed a model to identify malicious packets injected into the vehicle Controller Area Network (CAN) of the vehicle.

The dataset was obtained by the Open Car Test-bed and Network Experiments generator (OCTANE).

IoT home environments were examined in [7]. DoS attacks for TCP/IP networks and Denial-of-Sleep attacks for wireless network present the corresponding vulnerabilities of the IoT gateways. Following the analysis of packets captured from the PPP interface, samples of statistical data were extracted to synthesize the dataset and train a dense RNN.

DDoS attacks taking place in Software Defined Networking (SDN) is the target of [50]. Attacks took either on the data or on the control plane. Multiple sparse auto-encoders are connected to each other to construct a Stacked Auto-Encoder (SAE) leading to the classification of real network data derived from private network testbed.

In [12], A. A. Diro et al. designed and implemented a distributed attack detection mechanism based on DL methods. Network implementation assumed Fog Computing architecture, having a coordinating master-node responsible for collaborative parameter optimization. Training was performed in the Fog Nodes over the local traffic and inputs weights and biases were updated in parallel. The outputs were shared through a master node for global update and re-propagation.

As per [37] Deep Auto-encoders were used to detect the executed malware behavior. For each IoT device they constructed an auto-encoder model and refine training parameters and hyper-parameters, to achieve optimal results to unknown traffic behavior.

2.3 Graph-based approaches

An initial effort in this area of research was acknowledged in [54], where anomaly detection in web traffic was realized. Patterns of normal and abnormal behaviour were extracted out of servers' utilization. In this study graphs are modeling web requests and connections to the servers, whereas two features *malicious-server-degree* and *abnormal-traffic-score*, are extracted to point out probable malicious-clients intending to cause botnets or worm attacks.

A hybrid DL model constructed of Spectral Clustering and Deep Neural Network (SCDNN) was presented in [35]. Authors proposed spectral clustering to categorize KDD-CUP99 and NSL-KDD trainable datasets, to capture the significant features and diminish overall data processing complexity. Following dataset clustering into two to six groups, Ma, Tao et al utilized Denoising Auto-encoders (DAE), to represent the features with a level of distortion compared to the input pattern.

Dynamic graph anomaly detection was performed in [62], where an Attention-based temporal Graph Convolutional Network (GCN) model was developed. In this study, anomalous edges of the graph were identified utilizing temporal features as the long and short term patterns occurring within dynamic graphs.

Semi-supervised classification also involved the use of the GCN in a bordering approach proposed in [30]. Although the scope of this study did not

involve anomaly detection, the approach of graph-structured data with hidden layer representations of both the graph formation and the nodes' features was a basis for several research studies.

NetWalk [61] targets anomaly detection in dynamic networks. Following previous studies, they encoded the nodes of the network using vector representations and employed deep auto-encoders in conjunction with clustering techniques, thus managing to detect anomalies in real-time and address the network's continuous evolving.

A novel mathematical framework was proposed in [5], where relational inductive biases in the field of AI and deep learning, were studied and several different approaches concerning vision, language, control and decision making were unified. A common way to structure relations in a graph is proposed to assist learning about entities and their interconnections. To this regard, the graph network is the new building block proposed by the authors, extending current GNN approaches, posing a way to manage structured data and their interrelations.

Anomaly detection using GNN was also the main contribution proposed in [8]. In this study, outliers, noise or deviations from the normal status are considered an anomaly. The adjacency matrix of the involved nodes is computed to depict their interconnections. Detection mechanism is based on topological characteristics of the graph such as *between centrality, Degree*, and *Closeness*.

Anomalous edges detection is also proposed in [14] and [46], where the density of sub-graphs is taken into account in order to determine a pattern of malicious edges. In the former study sudden deviations in the density of the sub-graphs are forming a potential anomaly, concerning structural, temporal and content features of the sub-graph, whereas in the latter, a density function was defined in dynamic bipartite graph which utilized a greedy search mechanism and structural graph features to perform anomaly detection.

A node-level anomaly detection model was proposed in [4]. In this study the anomalous event is believed to generate a "clique" among the involved nodes. Thus, the authors introduce a probabilistic approach of detection, to infer the conditional probabilities of a "clique" generation in order to score a value of anomaly to each node. Real-world and synthetic sensor network data were utilized for the evaluation purpose. In [58] an insightful survey has been conducted resulting in a valuable taxonomy of GNNs mechanisms applied in several application domains. A detailed overview of Recurrent, Convolutional, Auto-encoders, and Spatial-Temporal Graph neural network approaches was presented. The need to model the appropriate GNN model to have high classification accuracy and F1 score is justified, where several GNN models are evaluated against the same benchmark datasets. This is also evident [18] where DARPA'98 and KDD'99 benchmark datasets are used to evaluate a Hybrid Deep Learning-Based Model for Cloud Data-center Networks anomaly detection, against state-of-the-art approaches. The article illustrates the method's out-performance againsth the comparative approaches, which in fact proves that adopting the optimal anomaly detection model results in superior detection accuracy values. Another study mentioning the significant added value of

GNN's adoption is conducted in [59], where powerful discriminative capabilities of GNNs are discussed.

2.4 Resource-efficient anomaly detection approaches

In [34] the limitations posed by the IoT resource-constraint environment are examined along with the anomaly detection proposed mechanism. They acknowledge the issue caused by sending all transformed data to a central Cloud, which consumes huge energy and bandwidth. Admittedly, designing low resource anomaly detection approaches, while remaining competitive in terms of accuracy is a challenge to overcome in the case of IoT networking. Towards this end, [43], proposes a light-weight detection mechanism for IoT sensor devices, while [49] makes an effort towards implementing a low-latency and high throughput deep packet anomaly detection technique. Another state-of-the-art approach addressing large memory, long training time and high classification latency is studied in [55]. A Random Forest implementation utilizing Decision Tree-Based Ensemble Methods is attempting to alleviate drawbacks caused by the anomaly detection procedure in the most efficient way.

Towards optimizing the anomaly detection techniques, several significant approaches have emerged. In [2] an extension to TEDA, a data analytics framework based on typicality and eccentricity, is presented. The authors propose a new condition for detecting anomalies which is independent of prior consumptions, as defined in TEDA. Within the proposed "σ gap" principle, the outlier is defined as a data point/sample that stands out and is different from other data samples. Based on this condition, data samples are analyzed with respect to their eccentricity and not on their values, thus a "σ gap" in terms of data samples' eccentricity, indicates an outlier.

3 Graph Neural Networks over a Multi-Agent System

3.1 Overal approach

An IoT network might comprise thousands of heterogeneous devices communicating over the Internet and being potentially utilized by millions of different end-users. Such a network should be considered under direct threat of an imminent attack at any time. Therefore, a method which shall provide security through the cooperation among IoT infrastructure devices can lead to quicker and more effective threat detection and mitigation results. We propose an extension to the current architecture approaches, which can help achieving security in the IoT network infrastructure while ensuring low latency results by exploiting the merits of an in-graph representation.

The main contribution stands on the distributed nature of a Multi-agent System (MAS) empowered with the GNN technology. As shown in Sec. 2.3 the graph-network is able to recognize the complexity of relations among nodes optimally, representing their inter-dependencies in a powerful way.

In the proposed approach, IoT network structures engage GNNs to adapt to the dynamics of the referred agent graph and offer higher level of cooperation as well as sophisticated learning strategies when compared to existing NN methods. Observations are exchanged only between neighboring agents. In this regard, globally shared information, as achieved through a central authority, is not required. Hence, requirements for bandwidth and computational power are reduced. Key factor to the current effort is the ability to accurately detect attacks and malicious events and avoid their impact based on both the node's and its neighboring nodes' view of the environment. The integration of a NN model alleviates the burden of processing large amount of data within the IoT environment, in terms of edges' and nodes' content and structural features.

The MAS extends hybrid NN implementations, which emphasized only on detection results. We add value with low latency anomaly detection and incorporate the distributed performance achieved by the communication between directly attached nodes. Unlike similar existing techniques, the injection of a graph to represent the network's links and also depict the agents, offers unique effectiveness concerning fast retrieval of the attack incident even in cases when a number of nodes has been compromised.

We also try to fill the gap of GNN application field. GNNs' have not been broadly used in IoT networking environments and to the best of our knowledge have not been used to identify malicious sources or predict upcoming attack events. A network of IoT devices can be easily seen as a graph consisting of node devices and communication channels between them as edges while a GNN can directly operate on a graph structure. Exploiting this, we can extract many of the features that appear in the network and use them to classify nodes. Essentially, every node is associated with a label which describes the behaviour of the node in the graph. This representation of information held the initial approach. Considering SDN edge and core forwarders as nodes belonging to the graph and their underlying connections as the graph's edges presents a natural way to achieve message passing over the interconnected entities. To summarize the key points of the proposed approach:

- Local monitoring of nodes and attached devices exploiting the underlying graph structure with the use of GNN networks is performed.
- –The complexity of IoT networks and the representation of the devices' interconnections in a formative, visual manner, suitable for various IoT application areas, is accommodated.
- Distributed anomaly detection ensuring lower bandwidth and power consumption measurements added by the proposed method, also addressing the distributed nature of attack patterns is performed.
- GNNs in the application area of anomaly detection in the IoT networking ecosystems are introduced.
- Extensive experiments on the IoT network data to prove effectiveness of the proposed model against traditional and State-of-the-art methods are conducted.

– Synthetic datasets based on real-world online retrieved data, to accommodate the training and evaluation needs of the proposed algorithm, are generated.

3.2 Intelligent agents positioned in the IoT network

This section describes the IoT network infrastructure, upon which the current proposal is based. We describe a general IoT network infrastructure to support the MAS implementation, which could potentially fit to several IoT application domains. Upon this matter we elaborate further in Sec. 5.

We consider a three-layered IoT infrastructure consisting of IoT devices, Fog Nodes and the Cloud. As in most cases of IoT, management applications dealing with time-sensitive data are running on the Fog Nodes, closer to the edge devices, avoiding the delays of transmitted vast amount of data to a central computational center. This in fact argues in favor of two distinct implementation scenarios of the MAS mechanism proposed. Therefore, implementing the agents either on the side of the Fog Nodes or on the SDN Edge Forwarders is considered. In the former case, the agent is seen as an application running on a virtual server or a container, while in the latter, Edge forwarders are able to support the agents' applicability with a dedicated AI processor of low-power. The SDN substrate in all cases is responsible for routing packets among the network entities, as core and edge forwarders are linked to the SDN control plane. A crucial aspect, which needs to be highlighted, is the localization of the traffic monitoring which is realized on each agent. Each agent is concerned with a subset of the overall transmitted traffic, including its directly attached agents, that is its neighborhood.

An example of the proposed architecture is depicted in Fig. 1, consisted of IoT devices and Edge/Core forwarders. Forwarders are hosting the AI agents in order to enable traffic monitoring , deriving from the IoT devices. SDN controller passes to the agents the traffic flows as raw data using a known standard for traffic monitoring. Within the scope of the proposal, NetFlow/IPFIX statistics are considered the optimal enabler of network management. Having acquired the input data to feed the intelligent agents, we proceed to the Feature Extraction strategy.

3.3 Input data

As explained, flows of raw network traffic are fed to each agent. Therefore, pre-processing of the raw information is required, for relevant features to attribute the underlying graph's edges and vertices. Raw data flows comprise several fields, describing network statistics and giving an insightful view of relevant information regarding protocols, IP addresses and their associated ports, along with more inspecting information as the total number of transferred bytes, packets, the duration, and the start time of each flow. With the

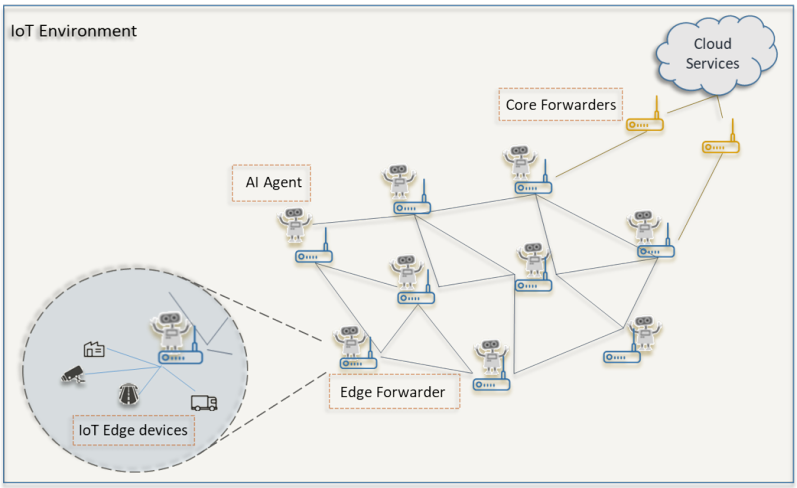


Fig. 1 Multi-agent system architecture, comprising SDN forwarders as part of an IoT networking example, attached to corresponding IoT devices and intelligent agents associated with each of the SDN forwarders for anomaly detection employment.

term flow, we generally address the bidirectional connections among the IoT nodes. Examples can be easily collected by online IoT related datasets, widely used to train and test several anomaly detection algorithms.

Within the current approach, there is a certain requirement to extract features, in order to assign attributes to the nodes and the edges of the network, which in fact poses the need to reconstruct flow-level information into node-level. This activity should result in incorporating features directly related to anomalous characteristics, omitting all non-relevant noise. To this regard, we highlight, the requirements of the proposed anomaly detection method, in terms of input data. Datasets need to include information with regard to the:

- **Start time**, which refers to starting timestamp of each connection (date-time object)
- **Duration** of each link measured in ms (int object)
- **Protocol** under which the connection is established (str/int object)
- **Source device** initializing the communication (str object)
- **Source IP** address (str object)
- **Target device** receiving the transmitted data (str object)
- **Destination IP** IP address (str object)
- **Direction** of the link (str object)
- **Total packets** transmitted over the link (int object)
- **Source bytes** transmitted (int object)
- **Total bytes** transmitted over the link (int object)
- **Label** normal/abnormal annotation of the connection

General properties should exist, as well. The number of nodes participating in the network is expected to be large, in order to serve the purpose of

graph formation and attribution. Various attacks should be addressed such as DDoS, Eavesdropping, Manipulation of IoT devices, Malware, Botnet etc. The temporal extend of the dataset, which denotes the overall length of the monitoring traffic time, is expected to be long, so as to assist in the creation of normal and abnormal pattern distribution.

3.4 Feature extraction methodology

To extract relevant features which are correlated with the existence of abnormalities we utilize the information referring to *Timestamp, Source, Destination, Number of Packets, Number of Bytes* , as well as *Direction* values. Information exchange in the dataset is depicted in the form of consecutive flows, in which source and destination entries are repeated describing numerous connections. To assign features specifically on each node and edge of the graph, we group the statistics to provide per node information, thus constructing a feature vector for nodes and for edges, separately.

Assuming j is the graph object we focus on, and i the specified time window, we define a feature vector as $r_i^j = [ps_i, pr_i, bs_i, br_i, dur_i]$, containing records with multiple indices, where j can be either a node (r_i^{node}) or an edge (r_i^{edge}).

The vector contains the following records; 'ps' (packets sent) that defines the number of packets sent from one node/edge to another in a certain period of time and 'pr' (packets received), defining the number of packets received. Moreover, the value 'bs' (bytes sent) defines the number of bytes sent from one node/edge to another in a certain period of time, while 'br' (bytes received) shows the number of bytes received from a node. Furthermore, the value 'dur' (connection duration) depicts the connection time in which two nodes/edge exchange data and a timestamp which follows the Datetime format. Finally, $\Delta t \in [t_1, t_2]$, $R_{\Delta t}$ is defined as $R_{\Delta t} = \{r_i^j V timestamp \in \Delta t\}$, and N is defined as $N = length(R_{\Delta t})$. Tables 1 and 2 contain the extracted groups of features (nodes' and edges') from raw network data, along with their mathematical explanation.

3.5 Integrating Agents with Graph Neural Networks

A graph is commonly used to process structural data, considering their interrelations. A significant benefit of this topological distribution of load is in fact the reduction in data processing activity required for each node. However, despite this gain, computational complexity expands in cases needing to include data of a prior and subsequent neighbor into the processing task. In an attempt to address this challenge, and enhance message passing process, Graph Neural Networks were introduced suggesting the representation of a node's vector as aggregated and transformed feature vectors of its neighbors.

Table 1 Node Feature Vector

Avg. number of Packets sent (aps)	$f_{aps}^{\Delta t}(j) = \frac{\sum_1^N ps_i}{N}$ f_{aps} is the average number of packets sent $ps \in r_i^{node} \forall r_i^{node} \in R_{\Delta t}$
Avg. number of Packets received (apr)	$f_{apr}^{\Delta t}(j) = \frac{\sum_1^N pr_i}{N}$ f_{apr} is the average number of packets received $pr \in r_i^{node} \forall r_i^{node} \in R_{\Delta t}$
Avg. number of Bytes sent (abs)	$f_{abs}^{\Delta t}(j) = \frac{\sum_1^N bs_i}{N}$ f_{abs} is the average number of bytes sent $br \in r_i^{node} \forall r_i^{node} \in R_{\Delta t}$
Avg. number of Bytes received (abr)	$f_{abr}^{\Delta t}(j) = \frac{\sum_1^N br_i}{N}$ f_{br} is the average number of bytes received $ps \in r_i^{node} \forall r_i^{node} \in R_{\Delta t}$
Avg connection duration (acd)	$f_{acd}^{\Delta t}(j) = \frac{\sum_1^N dur_i}{N}$ f_{acd} is the average connection duration $dur \in r_i^{node} \forall r_i^{node} \in R_{\Delta t}$

Table 2 Edge Feature Vector

Avg. number of Packets sent (aps)	$f_{abs}^{\Delta t}(j) = \frac{\sum_1^N bs_i}{N}$ f_{abs} is the average number of packets sent $bs \in r_i^{edge} \forall r_i^{edge} \in R_{\Delta t}$
Avg. number of Bytes sent (abs)	$f_{aps}^{\Delta t}(j) = \frac{\sum_1^N ps_i}{N}$ f_{aps} is the average number of bytes sent $ps \in r_i^{edge} \forall r_i^{edge} \in R_{\Delta t}$
Avg connection duration (acd)	$f_{acd}^{\Delta t}(j) = \frac{\sum_1^N dur_i}{N}$ f_{acs} is the average connection duration $dur \in r_i^{edge} \forall r_i^{edge} \in R_{\Delta t}$

The principles of Graph Networks GNN operation are described below, along with explanation regarding how GNN were adopted within the proposed methodology. Specifically, taking advantage of the GNN natural ability to separate distinct inputs, as repeatedly mentioned in literature, along with their capability to achieve message passing among the graph's nodes, this mechanism proves to be a competitive classification mechanism and an ideal way to represent relations.

In the MAS, cooperation among agents is represented by their inter-relations, which in fact exploits adequately GNN's inherent operation on a graph structure.

The basic computational block takes as input a graph and performing computational processes among the graph's nodes and edges, outputs a new graph instance. Let us denote nodes and edges which are the basic elements of the graph as v_i, and e_k, respectively.

We define a *graph* G as the tuple $G = (V, E)$ where $V = \{\mathbf{v}_i\}_{i=1:N^v}$ is the set of nodes (of cardinality N^v), and each $\mathbf{v}_i$ is a node's attribute while $E = \{\mathbf{e}_k\}_{k=1:N^e}$ is the set of edges (of cardinality N^e), where each $\mathbf{e}_k$ is the edge's attribute.

We adopt a modification of the framework constructed in [5] in order apply the proposed approach. To address message passing between network nodes, GNNs suggest the representation of a node's vector as aggregated and transformed feature vectors of its neighboring nodes. Thus, iterations of this aggregation procedure capture the structural information within the network neighborhood including neighbors in a n-hops distance, where n is the number of iterations. Thus, the computational processes within a graph-block include "update" functions ϕ, and "aggregation" functions, ρ, (r, s denote receivers and sender node indices)

$$\begin{aligned} \mathbf{e}'_k &= \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k},\right) \\ \mathbf{v}'_i &= \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i,\right) \\ \bar{\mathbf{e}}'_i &= \rho^{e\to v}\left(E'_i\right) \end{aligned} \tag{1}$$

where $E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{r_k=i,\ k=1:N^e}$, $V' = \{\mathbf{v}'_i\}_{i=1:N^v}$, and $E' = \bigcup_i E'_i = \{(\mathbf{e}'_k, r_k, s_k)\}_{k=1:N^e}$, ϕ^e computes edge-level updates whereas the ϕ^v computes node-level updates. ρ aggregates edge attributes per node. f determines what information is required as input.

$$\begin{array}{lll} \phi^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) & f^e\left(\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right) & = \mathrm{NN}_e\left(\left[\mathbf{e}_k, \mathbf{v}_{r_k}, \mathbf{v}_{s_k}\right]\right) \\ \phi^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i\right) & f^v\left(\bar{\mathbf{e}}'_i, \mathbf{v}_i\right) & = \mathrm{NN}_v\left(\left[\bar{\mathbf{e}}'_i, \mathbf{v}_i\right]\right) \\ \rho^{e\to v}\left(E'_i\right) & & = \sum_{\{k:\, r_k=i\}} \mathbf{e}'_k \end{array} \tag{2}$$

NN function used in Eq. 2 can be implemented through several neural network architectures. Exploiting this opportunity, we implement a fully connected Multi-layer perceptron (MLP), to perform message passing from the edges to their corresponding node and subsequently to update nodes' self attributes, respectively. The MLP architecture is chosen for faster execution and high accuracy results. We consider information exchange comprising succeeding steps and applied for each individual agent:

1. Identify of neighboring agents and connections.
2. Update e_1,e_2,e_3 edges' attributes applying Neural Network (NN_e) taking as input each edge along with its corresponding nodes, generating the updated edges, e'_1,e'_2,e'_3.
3. Update node's v_1 attributes applying Neural Networks (NN_v) on e'_1,v_1 inputs, generating the updated node v'_1.

Having exchanged information among neighboring agents, each agent uses the collected data to train and test the anomaly detection algorithm anomalies. Two types of anomalies are considered:

– The SDN forwarder is anomalous now, i.e. at time t
– The SDN forwarder will be infected by a malware that propagates in the network, i.e. infected at time $t + \Delta T$

3.6 Graph Multi-Layer Perceptron model

This section elaborates on the architecture of the Agent's GNN mechanism. We developed two separate models, for updating the nodes' and their corresponding edges' attributes, therefore introducing two consequent MLPs. Each of them results in classifying the nodes' and edge's status respectively. Hence, we succeed in implementing a communication channel and establishing a neighborhood of agents and information exchange within it. The number of input neurons represents four attributes of the edge's MLP or five attributes of the node's respectively. As the MLP belongs to the Feedforward Neural Network (FNN) family, neurons are connected via one-way links.

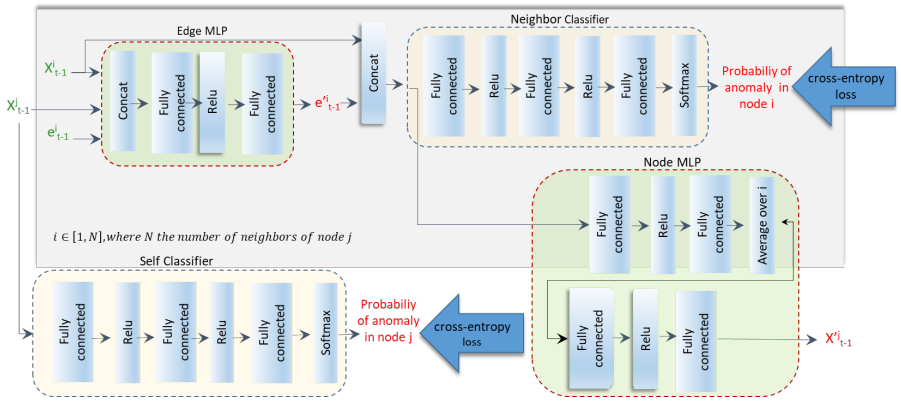


Fig. 2 GraphNET - GNN architecture employed on each Intelligent Agent

Fig. 2 gives a detailed view of the proposed GNN architecture, which is employed by each agent. Core blocks of the architecture are the edge and node MLPs.

The *edge MLP* is used to classify features and predict the probability of an anomaly on neighboring nodes whereas *node MLP* is used for node's features update and results in the probability of his own abnormal status.

Examining in detail each block, we define input and output. Assuming that we have node j and its neighboring nodes $i = 1, 2 \ldots, N$, edge's MLP input consist of the feature vector of a neighbor (denoted as X^i_{t-1} at time $t-1$), his own information (X^j_{t-1}) along with neighbor's corresponding edge features vector (denoted as e^i_{t-1}). Feature vectors are explicitly described in Tab. 1, 2. Concatenation is required before passing the data through the fully connected layers to output the updated edge feature vector (e'^i_{t-1}). In the next step, the updated edge vector is used as input to a softmax classifier along with the node's i features. After their passing through fully connected layers the probability of an anomalous node i (neighbor) is computed.

The node's MLP similarly is responsible for updating node's own feature representation based on the collected information. Concatenation result of Edges' MLP (e'^i_{t-1}) and the node's own feature vector is passed through the fully connected layers, and is subsequently classified to produce the probability of node's j (individual node) anomaly. We achieve exchange of information in an effective way justifying the value of the multi-agent system compared to other centralized structures, in an attempt to achieve detection of attacks and their distribution over the network.

We trained the *GraphNET*, the proposed structured model, using Adam optimizer [29] and the Binary Cross Entropy function [42]. We trained the model in four different experiments against several scenarios, derived from the generation of synthetic datasets, obtaining high accuracy results that can be shown in the following diagrams (Sec. 4.3). To produce normal and abnormal occasions out of the resulted probabilities, we considered a threshold value of 0.50. Using Pytorch [41] and Pytorch Geometric [15] libraries, the GNN implementation was performed with Python.

4 Performance evaluation

In this section we present experimental results for the *GraphNET*.We also describe the generated datasets. In a separate subsection we elaborate on the estimated values of bandwidth and power consumption overhead caused by the proposed method, in comparison to several Machine Learning algorithms and State-of-the-art methods developed.

At this point of implementation, we consider utilizing a pre-trained model, which having learned several patterns of network traffic is able to detect irregularities. However, the need to address the continuity of network traffic data can be addressed with the implementation of a buffer. This mechanism can hold incoming streams during training.

4.1 Synthetic datasets generation

Although numerous online datasets, depicting normal and anomalous network traffic of various attacks (Malware, Spam, ClickFraud, Port Scan, UDP/ICMP DDoS, Command and Control P2P connection, HTTP traffic etc.) were retrieved, structural data to represent graph-based information were not sufficiently gathered. Despite presenting realistic networking scenarios, the aforementioned datasets lacked fields referring to the source, destination, traffic statistics, starting time of the connection or did not include annotated data. This was the case in CICIDS 2017[45], UNSWNB15[39], ISCX-IDS-2012[47], NSL-KDD[51], ISCX [6], ADFA [11].

To address this issue we generated data depicting structural relationships among IoT nodes and edges. We utilized CTU-13 [16] instances, after determining the data distribution over normal and abnormal patterns, in order to set the basis of the synthetic data generation. All datasets produced followed the same distribution, while differentiating the total numbers of packets sent, received and their duration to acquire a broader view of the proposed method's capabilities and experiment on various network traffic conditions. In Fig. 4.1, 4.1, 4.1 is presented the packets' distribution of the generated datasets, fitting to the real-world dataset's (CTU-13) packets distribution.

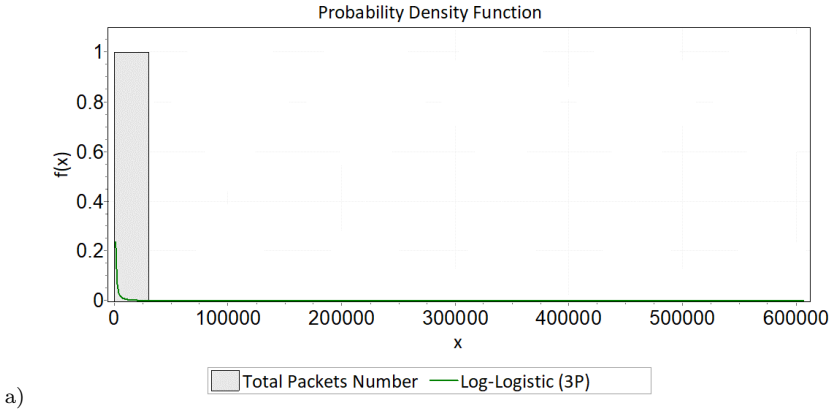
4.1.1 CTU-13 botnet dataset

To regenerate the exchange of data we thoroughly examined the present dataset, recorded in CTU University, Czech Republic, 2011. CTU-13 includes Botnet, Normal and Background traffic labels. It consists of 13 different attack scenarios. All cases define botnet instances sourced by a single malicious IP, that perform adversarial activities while exploiting various protocols. Each scenario was recorded in documents using the format .pcap [52] that was subsequently processed in order to obtain network flow data.

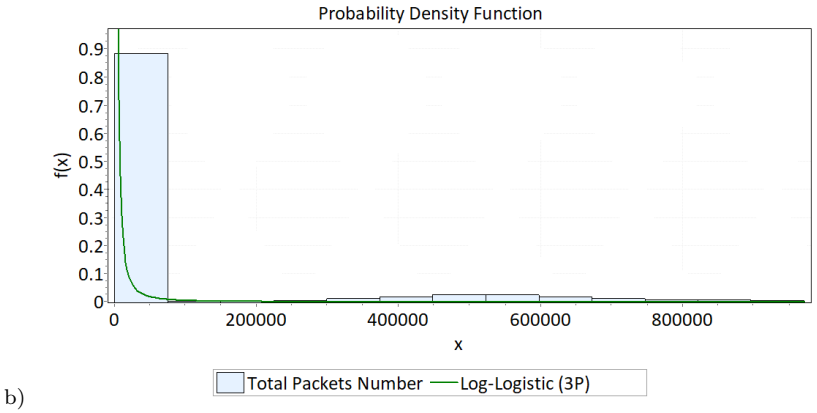
4.1.2 Generated worm attack dataset

We simulated a worm attack in the synthetic dataset.Its instances of network traffic flows are labeled as normal and abnormal. The network is made up of 30 forwarders and 170 IoT devices that exchange data following the distribution met in the CTU13, in which we injected abnormal flows of the Mirai [31], [20] botnet signature.

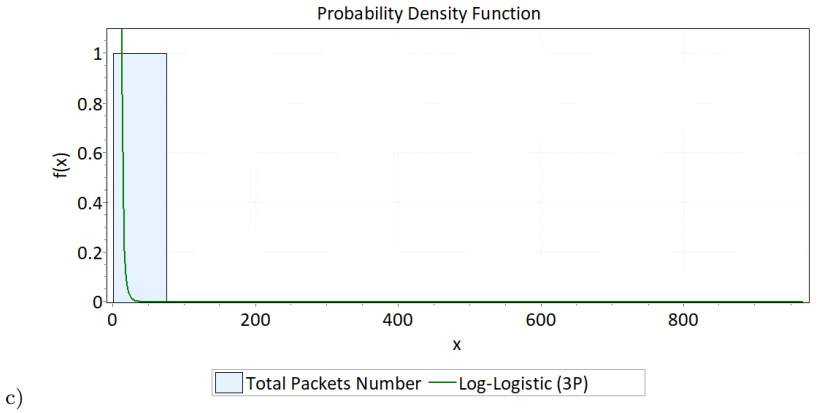
To better describe a DDoS attack taking place on an infected forwarder, we simulated offline state, resulting in null features vectors sent by maliciously affected forwarders. This also described the propagation impact. Examining this case of attack, we highlight the benefits of GNNs in detecting an attack before widely spreading in the network. An example is presented in Figure 4: where the agent of the edge forwarder has been attacked by a worm deriving from an IoT surveillance camera. Showing abnormal behaviour the malware



a)



b)



c)

Fig. 3 a) Total packets number of the propagating malware dataset, fitting in the Log logistics [3P] non-discrete distribution. b) Same metric of total packets number of the botnet CTU-13 dataset, fitting in the Log logistics [3P] non-discrete distribution. c) Total Packets number of infiltration attack dataset, fitting in the Log logistics [3P] non-discrete distribution.

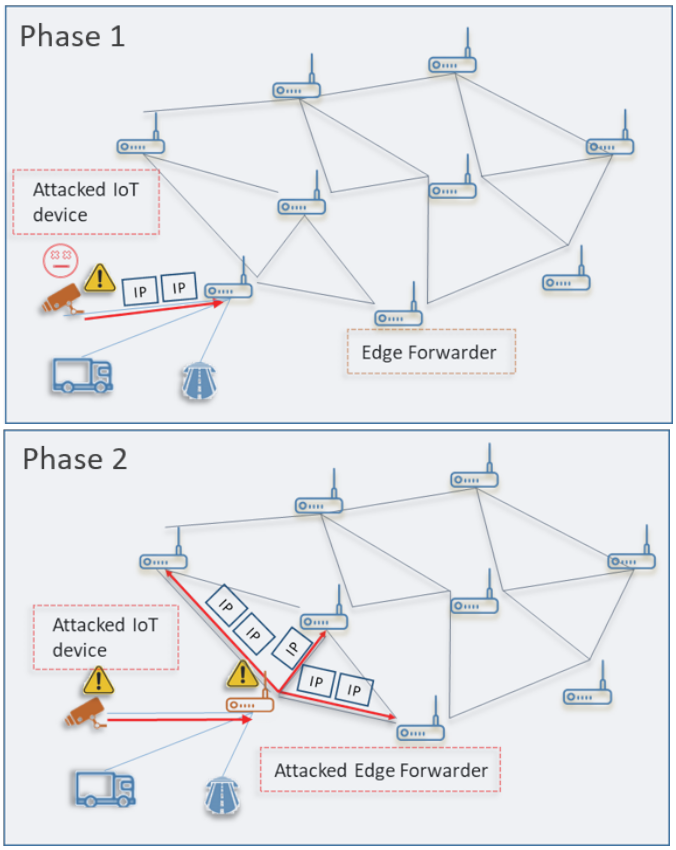


Fig. 4 Worm attack is initiated on a Surveillance camera (Phase 1) and is subsequently propagated to the next hop connected SDN Edge forwarder (Phase 2)

tries to propagate the infection further in the IoT network. Phase 2 of the propagation shows the contamination spreading to the attached forwarders.

4.2 Bandwidth and power consumption estimations

The results with respect to Accuracy, Roc AUC score, Bandwidth and Power Consumption are compared with those of centralised ML algorithms (SVM, Decision Tree, Random Forest), commonly used for anomaly detection, and two state-of-the-art implementations of Deep Auto-encoders proposed by [37], and a hybrid sequential proposed in [60], to showcase the improved performance offered by the proposed initiative.

We estimated the additional Bandwidth and Power consumption caused by the Anomaly Detection method in case of deploying a centralised and decentralised anomaly detection method. Having stated the number of SDN forwarders involved in the network traffic and the connection flows taking place

in every second we estimated based on the following mathematical equations [56].

Centralized case: $Bc = n * Fn * Fb + e * Fe * Fb$, where n is the total number of forwarders, Fn is the forwarders' feature number, Fb is the number of bits of a feature, e is the total number of communication links and Fe is the communication links' feature number Distributed case: $Bd = n * a * Fn * Fb$ where n is the total number of forwarders, a is the average number of forwarders' neighbors, Fn is the forwarders' feature number, Fb is the number of bits of a feature. Power consumption added by the anomaly detection method $E = Eidle + Einc, bit * B$, where $Eidle$ is the baseline energy consumption, $Einc, bit$ is the incremental energy per bit and B is the bandwidth.

4.3 Evaluation results

We compared accuracy of detection and ROC score depicted in the ROC diagrams presented in Fig. 5, and Fig. 6 and the corresponding tables. Two attack scenarios were examined referring to the worm infiltration and worm propagation attack. Three traditional ML classifiers and two state-of-the-art methods were trained and tested against the same datasets to be cross-evaluated along with the proposed method.

Results in Tables 3, 4, 5 and 6 show ROC score, accuracy, bandwidth and power consumption. In the corresponding diagrams, GNN's yellow curve proves out-performance compared to the residual methods. Two sub-scenarios were also taken under consideration, describing the event in which three abnormal forwarders go offline during the attacks.

The fact that the proposed method surpasses overall computed accuracy proves that the multi-agent cooperation and their information exchange is crucial, in order for the directly attached healthy nodes to acknowledge the undergoing attack on their neighbors. Distributed nature of the MAS is also yielding lower Bandwidth and Power Consumption results, since the feature vectors are only transferred between neighboring agents, through directly attached edges, thus avoiding the information burden and data processing on a single central node.

Tables 3 and 4, show that the GNN method presents the same high results of accuracy in both sub-scenarios of the infiltration attack, unlike the residual methods. In the case of the propagating attack, as depicted in Tables 5 and 6 the GNN method is also not affected by the offline compromised forwarders, proving that neighboring agents are aware of the abnormality. More specifically, in Table 3 detection performance of the proposed GNN model is illustrated. Although all methods achieve high accuracy results, it is evident that the proposed approach outperforms the comparative methods in terms of Bandwidth and Power Consumption. This effect is justified by the minimization of distributed feature vectors transferred, among neighboring agents, unlike the massive message transfer to a central computational unit, taking

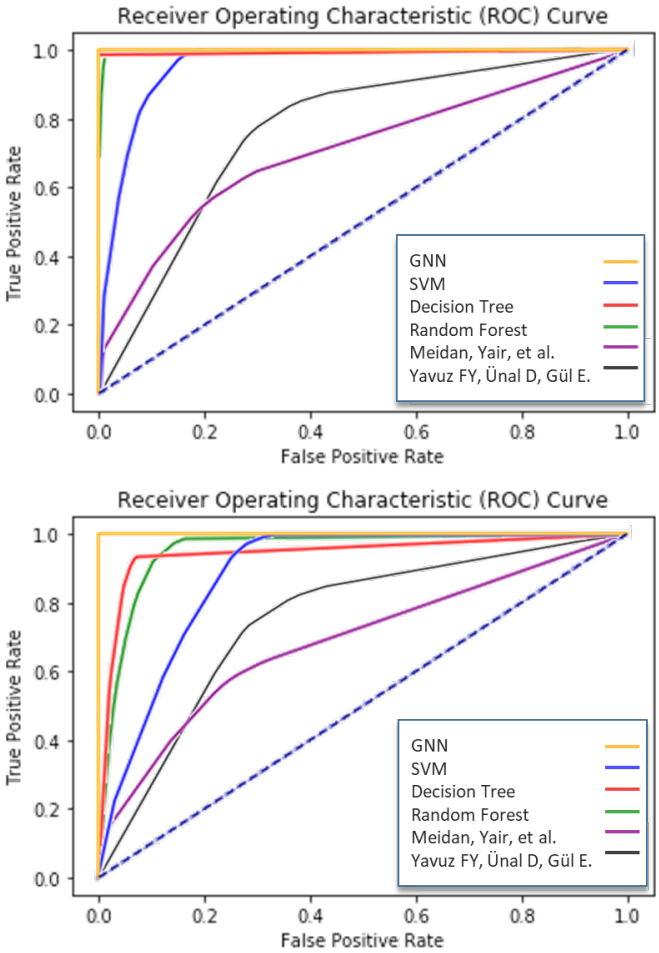


Fig. 5 Infiltration attack is initiated on a Surveillance camera and is propagated to the connected SDN Router

place in the centralized methods. This low-resource consumption is essential within the IoT environment.

Table 4 on the other hand shows out-performance of the proposed algorithm both in terms of resource consumption and accuracy results. The compromised forwarders have gone offline causing a drop in the transmitted traffic throughout the network, as illustrated in Bandwidth and Power Consumption values. However, the fact that a smaller number of monitors is now responsible for anomaly detection, drops the accuracy results for all centralized methods whereas GNN's inter-connected agents remain unaffected.

Similarly, in Table 5 the results regarding the attack propagation scenario are illustrated. Since intelligent agents are exchanging features and are able to classify the probability of neighbor infection, propagating attack is iden-

Table 3 Evaluation results against infiltration attack dataset

Method	Roc AUC score	Accuracy	Bandwidth (Kbits/s)	Power Consumption (Watts)
GNN	0,99	0,99	0,45	$1,53 * 10^{-6}$
Random Forest	0,99	0,99	33,98	$1,15 * 10^{-4}$
SVM	0,95	0,95	33,98	$1,15 * 10^{-4}$
Decision Tree	0,99	0,99	33,98	$1,15 * 10^{-4}$
Meidan,Yair, et al.	0,70	0,70	33,98	$1,15 * 10^{-4}$
Yavuz,FY, et al.	0,77	0,77	33,98	$1,15 * 10^{-4}$

Table 4 Evaluation results against infiltration attack dataset - three compromised forwarders have gone offline

Method	Roc AUC score	Accuracy	Bandwidth (Kbits/s)	Power Consumption (Watts)
GNN	0,99	0.99	0,40	$1,38 * 10^{-6}$
Random Forest	0,95	0,95	32,67	$1,11 * 10^{-4}$
SVM	0,88	0,88	32,67	$1,11 * 10^{-4}$
Decision Tree	0,94	0,94	32,67	$1,11 * 10^{-4}$
Meidan,Yair, et al.	0,68	0,68	32,67	$1,11 * 10^{-4}$
Yavuz,FY, et al.	0,75	0,75	32,67	$1,11 * 10^{-4}$

tified accurately by the GNN method. This is not the case in the remaining algorithms were the spreading of attack sources is not efficiently recognized. This is justified by the fact that abnormal nodes are considered normal in the initial learning phase, since the attack has not been spread, yet.

Table 5 Evaluation results against propagating Worm attack dataset

Method	Roc AUC score	Accuracy	Bandwidth (Kbits/s)	Power Consumption (Watts)
GNN	0,99	0,97	15,1	$5.134 * 10^{-5}$
Random Forest	0,96	0,96	10.800	$3,6 * 10^{-2}$
SVM	0,96	0,96	10.800	$3,6 * 10^{-2}$
Decision Tree	0,96	0,96	10.800	$3,6 * 10^{-2}$
Meidan,Yair, et al.	0,96	0,96	10.800	$3,6 * 10^{-2}$
Yavuz,FY, et al.	0,97	0,97	10.800	$3,6 * 10^{-2}$

Following the results of Table 5, in Table 6 accuracy scores drops further considering the centralized methods, while the GNN is again not affected by the DDoS caused by the worm propagation.

Simulated datasets presented diverse configurations. Therefore, the number of flows per dataset and the captured traffic duration are varying. This is depicted in the altered density of the attached edges among the nodes of the

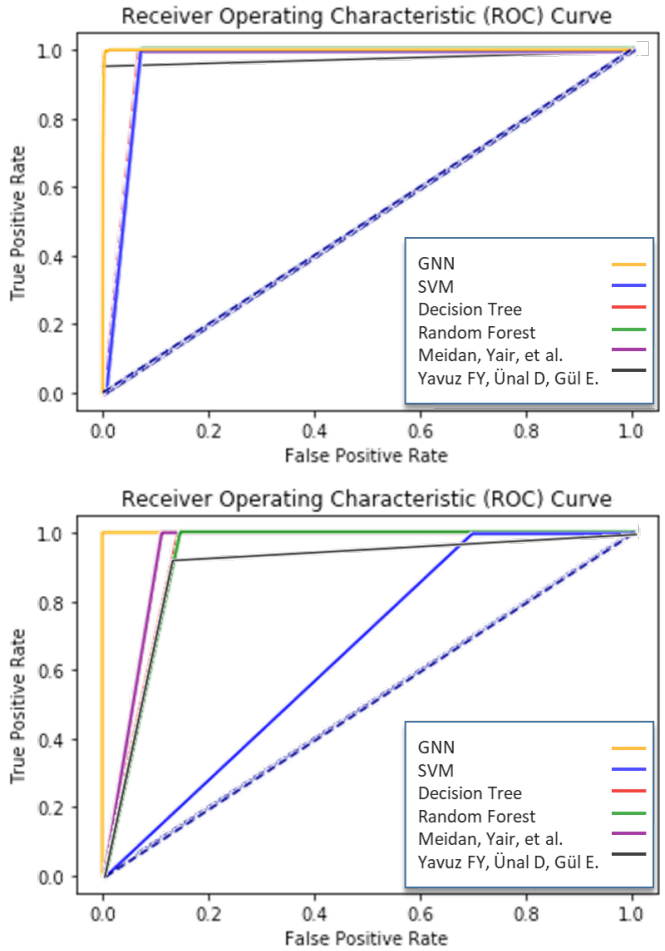


Fig. 6 Worm attack is initiated on a Surveillance camera and is propagated to the connected Forwarder

Table 6 Evaluation results against propagating attack Worm dataset - three compromised forwarders have gone offline

Method	Roc AUC score	Accuracy	Bandwidth (Kbits/s)	Power Consumption (Watts)
GNN	0,99	0,96	13,6	$4,624 * 10^{-5}$
Random Forest	0,92	0,92	10.400	$3,5 * 10^{-2}$
SVM	0,65	0,65	10.400	$3,5 * 10^{-2}$
Decision Tree	0,92	0,92	10.400	$3,5 * 10^{-2}$
Meidan,Yair, et al.	0,94	0,94	10.400	$3,5 * 10^{-2}$
Yavuz,FY, et al.	0,89	0,89	10.400	$3,5 * 10^{-2}$

network, per time-window, which in turn affects the complexity of the dataset and the time needed for the algorithm to be trained on it.

We have created two types of datasets describing two attack types, a distributed worm infiltration attack and a propagating case which follows the rootto-branches dissemination pattern. We extensively trained the developed algorithm against all sub-scenarios. Thus, in table 7 we present all gathered time measurements presenting the algorithms requirements, in terms of training against the different edges' sparsity and monitoring duration values.Training was successful for each dataset after a total of 200 epochs, along with parameterization of 0.01 learning rate and 5e-4 weight decay, selected after rigorous fine-tuning.

Table 7 Training execution time

Dataset type/version	Training time (sec)	Number of flows	Captured traffic total duration (sec)
Infiltration attack/1	90,54	19.135	5
Infiltration attack/2	89,69	17.693	5
Propagation attack/1	290,52	211.231	5
Propagation attack/2	168,92	40.463	300

Training time is raised significantly when the captured traffic has high duration, since the processing time to compute per node/edge features is increased. Data processing is more demanding when the number of flows mirroring the number of edges per node is increased, as well.

5 Discussion

This section presents several application domains, where the proposed method could be potentially deployed. We elaborate on hardware requirements and limitations posed by training the detection model on less flexible IoT network devices. Furthermore, we discuss extension paths to be followed to evolve the proposed algorithm.

5.1 GNN-based AI agents - Areas of application

5.1.1 Smart electricity grid

The case of electricity grid has raised great research interest, so as to fit the quality standards, posed by the US Department of Energy DOE's Modern Grid Initiative, and exhibit characteristics of self-healing, attack resistance, ability to accommodate storage options, market enabling, power quality meeting the modern needs, asset optimization and efficient operation. To satisfy

these requirements, a distributed multi-agent system is best suited to allocate different tasks concerning system voltage monitoring, energy resources and system information storing, and monitoring electricity consumption, as considered in [33], [48].

The need for resistance against cyber attacks, or the recognition of energy demand patterns and behaviors of the consumers, could be potentially met with the use of GNNs, which could best accommodate the large amount of transferred data and model the network of agents and their interactions efficiently.

Assuming a probable case of DDoS attack in the Smart grid, compromised edge devices intend to produce a UDP flood on a part of the network and therefore render it unavailable. With the adoption of the proposed solution, agents will be collecting traffic data and observations of the neighboring environments, implementing a GNN to detect and mitigate the attack pattern.

An agent-node receives data packets belonging to its neighboring node containing information about the traffic flow. A set of relevant features will be extracted (Number of destination UDP ports per second). A GNN layer is aggregating the feature vectors of the neighboring agents into an updated vector of features which is classified giving the probability of neighboring infection. Feature vector aggregated with the agent's own attributes, will be subsequently passed through the GNN's layers to classify the agent's own probability.

In such a complex IoT network structure, GNN is expected to both depict accurately the interconnections among the agents and to process the large amount of transmitted data in parallel, thus taking advantage of the local computational processes performed by each node. The proposed method, is the most suited approach to visualize structured data of such an IoT case, as proven previously by the adoption of learning with graphs in biological, and financial networks. An example of graph representation learning for biological networks depicting protein-protein inter-relations, is examined in [21].

5.1.2 Internet of Medical Things

Being a vastly growing area of interest, Internet of Medical Things (IoMT) is willing to enable affordable and reliable health care services through the interconnection of devices and the provision of real-time solution to medical issues. To offer high QoS to the customers and stakeholders, it is crucial to ensure the privacy, confidentiality, integrity of the transmitted data such as vital sign measurements, laboratory test results, procedures, images and imaging reports, blood pressure, respiration, oxygen saturation measurements and authentication of users, striving against physical and cyber-threats.

In a network of interconnected devices, medical equipment sensors and actuators are composing the lower level of the IoMT architecture [22]. Intermediate layer also referred to as Gateway or Middleware, is constructed to handle the interaction of the heterogeneous medical devices through several communication protocols. To process big data flows and facilitate information

sharing among the intermediate central devices, a distributed MAS implementation on the side of the intermediate layer devices, is certain to enhance remote users' collaboration.

Integrating this IoMT implementation with the proposed algorithm, we consider employing the network of distributed AI agents on gateway devices. We are considering the presence of an anomaly, such as a blackhole attack, under which an actuator is dropping control input packets. Traffic data are gathered to the monitoring agents, and distributed classification procedures will be followed as defined by the proposed model architecture.

5.1.3 Food Chain infrastructure

Food chain infrastructure comprises several intermediate processes and stakeholders. After being initiated at the food production points it proceeds to manufacturing, distribution and retail activities before ending up on the consumer's table. Integration of the Food Chain with the IoT resulted in major transformations in the Food Industry and assisted in the food safety, optimized logistics and enhanced QoS. Continuous monitoring of all sequential procedures of production, storing, shipping, and selling decreased the overall cost and risk and managed production quantities taking into consideration consumers' choices, to lower waste of products.

Given the fact that immediate response in emergency situations is cost-efficient, the implementation of a multi-agent system to run GNN on the IoT forwarders transmitting the information flow, is a promising scenario. The envisioned architecture includes IoT edge devices responsible for transmitting information helping the durability of food products. IoT forwarders transfer data streams across the network. Therefor choosing the lighter path can support the network's performance.

We assuming the network consists of multiple forwarding devices, on which AI agents are running. GNNs model the way agents are inter-connected.

5.1.4 Intelligent Transportation System infrastructure

Vehicles and other transportation infrastructures are interconnected in the case of ITS. Similarly to all IoT scenarios described, a security mechanism is a key aspect of its seamless function. The purpose of ITS is to integrate computers, electronic devices, sensors, actuators to realize the interconnection of vehicles and facilitate information exchange between them. Assuming that a security condition has been breached and a smart vehicle has violated a traffic light, it is a matter of safety for the residual interconnected vehicles to acquire this information. The urgent need for fast transmission describes the network requirement for low latency. In this case, we assume the implementation of AI agents to run on IoT sensor-hub devices. Information deriving from the IoT edge devices (road-side units) will be gathered to the sensor-hub monitoring agents. Distributed anomaly detection will be performed since neighboring agents will exchange their view of the environment's status.

5.1.5 Computer Vision domain

Anomaly detection regarding health image processing unveils another domain upon which tremendous progress has been made. In [3] an approach towards real-time detecting and tracking in video streams is presented. Specifically, an innovative method of video-analytics is proposed to process the frames in a one-pass manner using recursive calculations, thus canceling the requirement of information storage. The method introduces the recursive density estimation (RDE) technique for detecting the visualized object and proposes an online learning method based on the evolving Takagi–Sugeno (eTS) fuzzy systems to predict and update the position of the object in the video stream. In other approaches, hybrid models of Deep Learning have been combined to deliver the most effective detection techniques and simulate human's vision and logical reasoning in malicious pattern recognition. Numerous literature research studies have been conducted within this application domain [44], [9], [40].

On the other hand, computer vision domain includes image classification area which has been integrated with GNN mechanism in state-of-the-art research thrusts. Specifically in [17] a collection of annotated input images is fed to a GNN mechanism, proposing a generalized approach, of few-shot learning models. In [19], Neural Graph Matching Networks (NGMN) are studied proposing a novel framework that can be trained to recognize previously unknown 3D actions by generating and matching graphs. In addition, in [32] a novel GNN mechanism termed Prototype Propagation Network (PPN) is proposed. PPN is trained on few-shot tasks and coarse-label weakly-labeled data Developing a propagation mechanism it results in graph of prototypes attempting enhance the few-shot learning models. Finally, in [27], health image processing area in integrated with the GNN approach. The study proposes a BrainNet Convolution Neural Network (BrainNETCNN) to learn synthetic Diffusion Tensor Images (DTI) of preterm infants resulting in structural brain connectivity networks.

5.2 IoT hardware requirements

Along with the adoption of AI algorithms for anomaly detection, hardware requirements of the IoT devices are under inspection. That is to identify whether the IoT devices under learning and detecting activities, are in fact able to undertake the computational cost. In the case of Fog Nodes, where Virtual servers are implemented, several delay-sensitive applications are deployed. This solution is also adopted by autonomous vehicle infrastructure, where delays can be disastrous. As presented in [63], a lot of research has been conducted towards Edge and Fog side computing, following the enormous expansion of the IoT device, and the application of ML mechanisms to process the large amount of transferred data. To this end, the aforementioned study summarizes a number of low-power, State-of-the-art Machine Learning dedicated processors. At the same time, Intel is also developing customized processors for Deep Learning

activities. Furthermore, IoT sensor hubs, processing data closer to the edge, can support Machine Learning algorithms. [24] and [23] present NVIDIA's 7.5-watt supercomputer and Google's purpose-built ASIC both designed to run AI application on the edge.

6 Conclusion and future work

In this article, a method for the detection of anomalous events was presented. This method applies a GNN mechanism and fully connected networks to create an edge and node classifier resulting in the probability of infection on a node and its corresponding vertices. Exploiting the graph representation we achieved information exchange within an agent's neighborhood assisting in identifying abnormalities and security breach incidents based on their interrelations. We took advantage of the view that a node has with regard to his neighboring nodes' health status in order to solve the problem of inadequate number of monitors against distributed attack patterns, and compromised nodes being unaware of their infected condition. At the same time, we attempted to reduce resources such as the bandwidth and energy consumption compared to centralized IDS. Each active node applied GNNs to the feature values concerning the interconnections in its local neighborhood. For the GNN to be trained and tested, we used real-world dataset's normal traffic patterns and additionally we generated data based on the *Mirai's* abnormal distribution.

To set a solid basis for the proposed approach, in Sec. 2, we performed a broad review of the most recent methods with respect to anomaly detection, applied in separate IoT infrastructures, as in a centralised or decentralised or manner. Furthermore we studied several graph-based approaches with respect to different network infrastructures. Most of the methods used hybrid architectures to better process the acquired features, and the different use cases of network abnormal behaviour. The results proved the method is feasible to be used to ensure stability in network's seamless operation, showing high accuracy and low bandwidth and power consumption values against distributed attack scenarios. Experiments were conducted with the use of generated datasets based on real traffic distributions, to contain malware instances within normal traffic data. Three residual ML classifiers (SVM, Decision Tree, Random Forest) along with a two state-of-the-art approaches were also implemented to be trained and tested against the same generated data and thus evaluate the accuracy scores of the GNN proposed method.

Blocking potential intruders and cyber-criminals is conventionally achieved using encryption against unauthorized access on the edge devices, whereas network security requires data encryption protocols. Integrity of personal and other sensitive data is ensured with the adoption of suitable security policies. The detection of attacks, malicious patterns and irregular behavior, along with the required mitigation and avoidance mechanisms, has unfolded a promising topic of research.

6.1 Future work and extensions

Our future plans include several enhancements of the proposed method, with respect to early forecast of attacks, with the use of time-series temporal features, and experiments on graph-RNN Neural Network architecture. We plan to incorporate the neighbors' view of their environment, combining all opinions in way that not only directly attached, but also distant neighbors have a view on other nodes. Additionally, we plan to evaluate the proposed method against known IDSs and complete the security scheme with countermeasures and mitigation actions to ensure the IoT's system stability and fast recovery. We plan an effort towards employing the GNN anomaly detection mechanism on network testbeds, using Contiki and Mininet linux-based software, so as to measure accuracy, bandwidth and power consumption scores on real traffic scenarios were multiple IoT nodes are involved. Finally, in a separate scenario of implementation, the GNN method could be deployed in conjunction with a network infrastructure, such of Software Define Perimeter (SDP) [38]. Since SDP is a security architecture defining various modes of communication, (client-server, server-server, client-gateway, client-server-client) it could provide the GNN with network data to be analyzed. SDP is attempting to authenticate hosts before enabling communication with the presence of a controller, hence partitioning the network to avoid unavailability or a single-point-of-failure, is crucial. In that case, GNN distributed monitoring could complement the attack resistance promoted by the SDP, by employing a light-weight anomaly detection, locally on the inter-connected gateways.

Acknowledgements This work is supported by the European Unions Horizon 2020 Research and Innovation Program through the SerIoT project under Grant Agreement No. 780139 (https://seriot-project.eu/project/).

References

1. Al-Qatf, M., Lasheng, Y., Al-Habib, M., Al-Sabahi, K.: Deep learning approach combining sparse autoencoder with svm for network intrusion detection. IEEE Access **6**, 52843–52856 (2018)
2. Angelov, P.: Anomaly detection based on eccentricity analysis. In: 2014 IEEE symposium on evolving and autonomous learning systems (EALS), pp. 1–8. IEEE (2014)
3. Angelov, P., Sadeghi-Tehran, P., Ramezani, R.: An approach to automatic real-time novelty detection, object identification, and tracking in video streams based on recursive density estimation and evolving takagi–sugeno fuzzy systems. International Journal of Intelligent Systems **26**(3), 189–205 (2011)
4. Bars, B.L., Kalogeratos, A.: A probabilistic framework to node-level anomaly detection in communication networks. arXiv preprint arXiv:1902.04521 (2019)
5. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)
6. Beigi, E.B., Jazi, H.H., Stakhanova, N., Ghorbani, A.A.: Towards effective feature selection in machine learning-based botnet detection approaches. In: 2014 IEEE Conference on Communications and Network Security, pp. 247–255. IEEE (2014)

7. Brun, O., Yin, Y., Gelenbe, E.: Deep learning with dense random neural network for detecting attacks against iot-connected home environments. Procedia computer science **134**, 458–463 (2018)
8. Chaudhary, A., Mittal, H., Arora, A.: Anomaly detection using graph neural networks. In: 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), pp. 346–350. IEEE (2019)
9. Chen, X., Pawlowski, N., Rajchl, M., Glocker, B., Konukoglu, E.: Deep generative models in the real-world: An open challenge from medical imaging. arXiv preprint arXiv:1806.05452 (2018)
10. Cheng, M., Xu, Q., Lv, J., Liu, W., Li, Q., Wang, J.: Ms-lstm: A multi-scale lstm model for bgp anomaly detection. In: 2016 IEEE 24th International Conference on Network Protocols (ICNP), pp. 1–6. IEEE (2016)
11. Creech, G.: Developing a high-accuracy cross platform host-based intrusion detection system capable of reliably detecting zero-day attacks. Ph.D. thesis, University of New South Wales, Canberra, Australia (2014)
12. Diro, A.A., Chilamkurti, N.: Distributed attack detection scheme using deep learning approach for internet of things. Future Generation Computer Systems **82**, 761–768 (2018)
13. Erfani, S.M., Rajasegarar, S., Karunasekera, S., Leckie, C.: High-dimensional and large-scale anomaly detection using a linear one-class svm with deep learning. Pattern Recognition **58**, 121–134 (2016)
14. Eswaran, D., Faloutsos, C., Guha, S., Mishra, N.: Spotlight: Detecting anomalies in streaming graphs. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 1378–1386 (2018)
15. Fey, M., Lenssen, J.E.: Fast graph representation learning with pytorch geometric. arXiv preprint arXiv:1903.02428 (2019)
16. Garcia, S., Grill, M., Stiborek, J., Zunino, A.: An empirical comparison of botnet detection methods. computers & security **45**, 100–123 (2014)
17. Garcia, V., Bruna, J.: Few-shot learning with graph neural networks. arXiv preprint arXiv:1711.04043 (2017)
18. Garg, S., Kaur, K., Kumar, N., Kaddoum, G., Zomaya, A.Y., Ranjan, R.: A hybrid deep learning-based model for anomaly detection in cloud datacenter networks. IEEE Transactions on Network and Service Management **16**(3), 924–935 (2019)
19. Guo, M., Chou, E., Huang, D.A., Song, S., Yeung, S., Fei-Fei, L.: Neural graph matching networks for fewshot 3d action recognition. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 653–669 (2018)
20. Hallman, R., Bryan, J., Palavicini, G., Divita, J., Romero-Mariona, J.: Ioddos—the internet of distributed denial of service attacks (2017)
21. Hamilton, W., Ying, Z., Leskovec, J.: Inductive representation learning on large graphs. In: Advances in neural information processing systems, pp. 1024–1034 (2017)
22. Irfan, M., Ahmad, N.: Internet of medical things: Architectural model, motivational factors and impediments. In: 2018 15th Learning and Technology Conference (L&T), pp. 6–13. IEEE (2018)
23. Google. edge tpu (2018). URL https://cloud.google.com/edge-tpu/
24. Nvidia jetson tx2 module. (2018). URL https://developer.nvidia.com/embedded/buy/jetson-tx2
25. Javaid, A., Niyaz, Q., Sun, W., Alam, M.: A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pp. 21–26. ICST (Institute for Computer Sciences, Social-Informatics and ... (2016)
26. Kang, M.J., Kang, J.W.: Intrusion detection system using deep neural network for in-vehicle network security. PloS one **11**(6), e0155781 (2016)
27. Kawahara, J., Brown, C.J., Miller, S.P., Booth, B.G., Chau, V., Grunau, R.E., Zwicker, J.G., Hamarneh, G.: Brainnetcnn: Convolutional neural networks for brain networks; towards predicting neurodevelopment. NeuroImage **146**, 1038–1049 (2017)
28. Kim, J., Kim, J., Thu, H.L.T., Kim, H.: Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International Conference on Platform Technology and Service (PlatCon), pp. 1–5. IEEE (2016)

29. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
30. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. arXiv preprint arXiv:1609.02907 (2016)
31. Kolias, C., Kambourakis, G., Stavrou, A., Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017)
32. Liu, L., Zhou, T., Long, G., Jiang, J., Yao, L., Zhang, C.: Prototype propagation networks (ppn) for weakly-supervised few-shot learning on category graph. arXiv preprint arXiv:1905.04042 (2019)
33. Logenthiran, T., Srinivasan, D.: Computational intelligence and smart grid. Computational Intelligence-Volume II p. 202 (2015)
34. Lyu, L., Jin, J., Rajasegarar, S., He, X., Palaniswami, M.: Fog-empowered anomaly detection in iot using hyperellipsoidal clustering. IEEE Internet of Things Journal **4**(5), 1174–1184 (2017)
35. Ma, T., Wang, F., Cheng, J., Yu, Y., Chen, X.: A hybrid spectral clustering and deep neural network ensemble algorithm for intrusion detection in sensor networks. Sensors **16**(10), 1701 (2016)
36. Maimó, L.F., Gómez, Á.L.P., Clemente, F.J.G., Pérez, M.G., Pérez, G.M.: A self-adaptive deep learning-based system for anomaly detection in 5g networks. IEEE Access **6**, 7700–7712 (2018)
37. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., Elovici, Y.: N-baiot—network-based detection of iot botnet attacks using deep autoencoders. IEEE Pervasive Computing **17**(3), 12–22 (2018)
38. Moubayed, A., Refaey, A., Shami, A.: Software-defined perimeter (sdp): State of the art secure solution for modern networks. IEEE Network **33**(5), 226–233 (2019)
39. Moustafa, N., Turnbull, B., Choo, K.K.R.: An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Internet of Things Journal (2018)
40. Nair, T., Precup, D., Arnold, D.L., Arbel, T.: Exploring uncertainty measures in deep networks for multiple sclerosis lesion detection and segmentation. Medical image analysis **59**, 101557 (2020)
41. Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A.: Automatic differentiation in pytorch (2017)
42. Rubinstein, R.: A stochastic minimum cross-entropy method for combinatorial optimization and rare-event estimation. Methodology and Computing in Applied Probability **7**(1), 5–50 (2005)
43. Sedjelmaci, H., Senouci, S.M., Al-Bahri, M.: A lightweight anomaly detection technique for low-resource iot devices: A game-theoretic methodology. In: 2016 IEEE International Conference on Communications (ICC), pp. 1–6. IEEE (2016)
44. Seeböck, P., Orlando, J.I., Schlegl, T., Waldstein, S.M., Bogunović, H., Klimscha, S., Langs, G., Schmidt-Erfurth, U.: Exploiting epistemic uncertainty of anatomy segmentation for anomaly detection in retinal oct. IEEE transactions on medical imaging **39**(1), 87–98 (2019)
45. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP, pp. 108–116 (2018)
46. Shin, K., Hooi, B., Faloutsos, C.: M-zoom: Fast dense-block detection in tensors with quality guarantees. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases, pp. 264–280. Springer (2016)
47. Shiravi, A., Shiravi, H., Tavallaee, M., Ghorbani, A.A.: Toward developing a systematic approach to generate benchmark datasets for intrusion detection. computers & security **31**(3), 357–374 (2012)
48. Singh, V.K., Ozen, A., Govindarasu, M.: A hierarchical multi-agent based anomaly detection for wide-area protection in smart grid. In: 2018 Resilience Week (RWS), pp. 63–69. IEEE (2018)
49. Summerville, D.H., Zach, K.M., Chen, Y.: Ultra-lightweight deep packet anomaly detection for internet of things devices. In: 2015 IEEE 34th international performance computing and communications conference (IPCCC), pp. 1–8. IEEE (2015)

50. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for network intrusion detection in software defined networking. In: 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263. IEEE (2016)
51. Tavallaee, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pp. 1–6. IEEE (2009)
52. Tcpdump/libpcap public repository (2018). URL https://www.tcpdump.org
53. Thing, V.L.: Ieee 802.11 network anomaly detection and attack classification: A deep learning approach. In: 2017 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1–6. IEEE (2017)
54. Tran, M.C., Heejeong, L., Nakamura, Y.: Abnormal web traffic detection using connection graph. Bulletin of Networking, Computing, Systems, and Software **3**(1), 57–62 (2014)
55. Vargaftik, S., Keslassy, I., Ben-Itzhak, Y.: Rade: Resource-efficient supervised anomaly detection using decision tree-based ensemble methods. arXiv preprint arXiv:1909.11877 (2019)
56. Vishwanath, A., Hinton, K., Ayre, R.W., Tucker, R.S.: Modeling energy consumption in high-capacity routers and switches. IEEE Journal on Selected Areas in Communications **32**(8), 1524–1532 (2014)
57. Wang, W., Sheng, Y., Wang, J., Zeng, X., Ye, X., Huang, Y., Zhu, M.: Hast-ids: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access **6**, 1792–1806 (2017)
58. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
59. Xu, K., Hu, W., Leskovec, J., Jegelka, S.: How powerful are graph neural networks? arXiv preprint arXiv:1810.00826 (2018)
60. Yavuz, F.Y., Ünal, D., Gül, E.: Deep learning for detection of routing attacks in the internet of things. International Journal of Computational Intelligence Systems **12**(1), 39–58 (2018)
61. Yu, W., Cheng, W., Aggarwal, C.C., Zhang, K., Chen, H., Wang, W.: Netwalk: A flexible deep embedding approach for anomaly detection in dynamic networks. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 2672–2681 (2018)
62. Zheng, L., Li, Z., Li, J., Li, Z., Gao, J.: Addgraph: anomaly detection in dynamic graph using attention-based temporal gcn. In: Proceedings of the 28th International Joint Conference on Artificial Intelligence, pp. 4419–4425. AAAI Press (2019)
63. Zou, Z., Jin, Y., Nevalainen, P., Huan, Y., Heikkonen, J., Westerlund, T.: Edge and fog computing enabled ai for iot-an overview. In: 2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS), pp. 51–56. IEEE (2019)