Future Generation Computer Systems 110 (2020) 91–106

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

A new network forensic framework based on deep learning for Internet of Things networks: A particle deep framework

Nickolaos Koroniotis *, Nour Moustafa, Elena Sitnikova

School of Engineering and Information Technology, UNSW Canberra Cyber, University of New South Wales Canberra, Australia

ARTICLE INFO

Article history:
Received 22 September 2019
Received in revised form 14 March 2020
Accepted 21 March 2020
Available online 30 March 2020

Keywords:
Network forensics
Threat detection
Attack tracing
Deep learning
Particle swarm optimization

ABSTRACT

With the prevalence of Internet of Things (IoT) systems, inconspicuous everyday household devices are connected to the Internet, providing automation and real-time services to their users. In spite of their light-weight design and low power, their vulnerabilities often give rise to cyber risks that harm their operations over network systems. One of the key challenges of securing IoT networks is tracing sources of cyber-attack events, along with obfuscating and encrypting network traffic. This study proposes a new network forensics framework , called a Particle Deep Framework (PDF), which describes the digital investigation phases for identifying and tracing attack behaviors in IoT networks. The proposed framework includes three new functions: (1) extracting network data flows and verifying their integrity to deal with encrypted networks; (2) utilizing a Particle Swarm Optimization (PSO) algorithm to automatically adapt parameters of deep learning; and (3) developing a Deep Neural Network (DNN) based on the PSO algorithm to discover and trace abnormal events from IoT network of smart homes. The proposed PDF is evaluated using the Bot-IoT and UNSW_NB15 datasets and compared with various deep learning techniques. Experimental results reveal a high performance of the proposed framework for discovering and tracing cyber-attack events compared with the other techniques.

1. Introduction

With the proliferation of the Internet of Things (IoT) systems, IoT device-related IP addresses are widely linked to the Internet to offer daily services and tasks to end-users and organizations. The IoT industry has experienced accelerated growth over the last few years, with projections supporting the continuation of this increase [1]. In 2018, approximately 7 billion IoT devices were connected to the Internet, while in 2019 that number is expected to double to 14 billion devices. One of the most popular applications for IoT in terms of deployed devices is the smart home sector which had 663 million devices in effect in 2017. Applications for smart homes include among others, smart lights, fridges, ovens, thermostats and locks. Another application of IoT, on a larger scale has been planned for several European countries, and it is called the 'smart city'. Industrial, agricultural and health applications are also on the rise with automation, cost efficiency and precision being the most integral contributors to this trend. Some examples of devices from the healthcare sector include patient monitors, energy meters and imaging-related devices (X-ray machines) [1].

Even though IoT devices are preferred over conventional devices and systems, such deployments remain quite vulnerable to several attacking techniques taking advantage of both well-known and new attack vectors [2–4]. In the 2018 Symantec report of Internet security threats [4], the total number of attacks targeting IoT devices for 2018 exceeded 57,000, with more than 5,000 attacks being recorded each month. Attackers perform various hacking techniques, for example Denial of Service (DoS), Distributed DoS (DDoS), ransomware and other botnets attacks, for exposing IoT systems and their networks. Hackers execute several hacking scenarios to compromise vulnerable, unpatched, un-updated and/or unencrypted IoT devices to achieve their motives, such as corrupting IoT resources, stealing sensitive information that IoT devices often store or even use the compromised devices as infection vectors. Hackers sometimes seek to compromise the physical security of smart homes by hacking smart locks and garage doors [5].

Defending against such cyber-attacks is difficult, as there exist no commonly accepted standards for designing IoT devices [2]. This means that in an IoT deployment multiple protocols, such as MQTT, Zigbee and LoRa, could interact increasing complexity and heterogeneity [6–8]. Furthermore, new attacks which rely on zero-day exploits are often preferred by attackers, as they are not easily detected by most security countermeasures. Due to the heterogeneous norm of IoT deployments, developing an efficient network forensic solution demands depth-analysis for tracing and detecting attack vectors [2,9]. Typically, the network forensic process is segmented into several distinct phases, whereby each

* Corresponding author.
E-mail address: n.koroniotis@student.adfa.edu.au (N. Koroniotis).

https://doi.org/10.1016/j.future.2020.03.042
0167-739X/

phase defines the necessary preparation, analysis and actions of investigation [10]. These phases are identification, collection, preservation, examination, analysis and presentation. The first three stages, define the forensic analyst's access to the crime scene and their activities. Initially, sources of potential evidence have to be identified, after which data has to be collected in a way that enables the preservation of the data and the chain of custody. In the next two stages, the collected data is processed, so that relevant evidence can be located, after which the evidence is analyzed to make inferences about the cybercrime. Finally, the results are organized and further elaborated upon, so that they can be used in a court of Law.

Various forensic frameworks have been proposed to provide solutions to the acquisition problem but they do not consider the entire phases of investigation [7,11–13] . Most of them rely on a public ledger scheme, where diagnostic and communication data are shared between multiple entities, such as the police and insurance companies. The benefit of such a scheme would be that during an investigation, all relevant information could be readily available to the forensic experts, while its integrity would be guaranteed via digital signatures. However, most of the network forensic frameworks focus on data acquisition rather than considering the entire forensic process. These frameworks introduce some disadvantages such as the violation of privacy, as a user's information is distributed between the stakeholders, and the added complexity that these frameworks require. Furthermore, the frameworks focus on the preservation and collection phases of the investigation.

To investigate network-related incidents, several types of files are studied for tracing attack vectors that expose IoT systems. The easiest source of traces to access, which is often preferred, is traffic collection where network packets are recorded and stored in full packet capture files. By examining the files, an investigator can determine if an attack occurred and find all metadata related to attack traces such as timestamp and source of attack. There are two approaches: deep packet inspection and network flow analysis, to process and examine the files and discover attack behaviors [2]. Deep packet inspection focuses on the payload of the packet, allowing for an in-depth analysis of the traffic that is captured, which may be more accurate at detecting certain attacks although it may face challenges, like the payload being encrypted, a common occurrence in current networks, it requires extensive storage and accessing the payload may violate privacy laws. On the other hand, network flow analysis utilizes a summarization of the network traffic, where mainly statistical features are extracted such as connection speed, exchanged bits and timing data, in order to produce results. For the requirements of the research presented in this paper, network flow is preferred, as we suggest in this work [2].

A key characteristic of IoT devices is that they are constantly active. As such, performing network flow collection from an IoT's network results in excessive amounts of data. In order to perform analysis on the collected data, automated mechanisms are often employed to eliminate human error, with one such popular automated method being the utilization of deep learning. Through deep learning models, an investigator is capable of rapidly detecting patterns in the network data (packets, network flows) which indicate the occurrence of an attack [14–17]. However, for such a deep learning model to be used, it first needs to be trained. Training a model requires the utilization of data and the selection of values for the model's hyperparameters. Although both data and hyperparameters are important for the training phase, hyperparameter optimization, referring to the selection of optimal hyperparameter values, is crucial as it defines the abstract structure and training conditions of the model [18,19]. Considerable work has appeared over the years in the field of hyperparameter optimization [18,20–22], with the trend being a shift from the time consuming manual selection that may not yield optimal results, to more automated processes. Regardless, no single optimization method has been accepted as the preferred method by the research community, and as such, work is ongoing. It is necessary to develop network forensic mechanisms based optimization to timely investigate security incidents in IoT networks [2,23,24].

In this paper, we propose a network forensic framework, named Particle Deep Framework (PDF), based on optimization and deep learning. The proposed framework describes multiple phases of a digital investigation process, focusing on smart home deployments. It includes a method, based on Particle Swarm Optimization (PSO) for selecting the hyperparameter values for the deep neural network (DNN) and shows through acquired metrics, that the produced DNN achieves very high accuracy while minimizing false alarm rates. Furthermore, we compare the performance of the produced DNN with other deep learning models. For the evaluations and comparisons, we used the Bot-IoT [25] and the UNSW-NB15 [26] datasets that have many new attack vectors.

The main contributions of this paper are as follows. First, a new network forensic framework, named Particle Deep Framework (PDF), based on optimization and deep learning was proposed. Second is the utilization of an optimization method based on Particle Swarm Optimization(PSO) to select the hyperparameters of the Deep Neural Network (DNN). Thirdly, the evaluation and comparison of the performance displayed by the produced DNN with other classification models. For the evaluations and comparisons, the Bot-IoT [25] and the UNSW-NB15 [27] were used. Results indicate a great improvement in the performance of a neural network trained by using the PDF, with an increase in accuracy and reduction in false positive and negative rates.

The rest of the paper is organized as follows. Section 2 includes related research in the application of deep neural networks to network forensics. Section 3 presents our proposed framework, first discussing the framework itself on a theoretical field, then describing the experiments we conducted, in order to build the binary classifier which separates normal form attack traffic. Section 4 introduced the PSO algorithm, and how the particles propagate through the search space. Section 5, explains how the PSO was utilized for the estimation of hyperparameters of deep learning models. Section 6 illustrates the deployment of the proposed PDF in an IoT network. Section 7 presents the experimental setup and discusses the acquired results from the trained models. Section 8 gives the attack scenarios that can be identified by the PDF. In Section 9, the performance of an MLP trained by using the PDF is compared with other machine learning models. Section 10 discusses attack families, their statistics and detection rates. In Section 11, the advantages and limitations of the PDF are discussed. Finally, in Section 12 the conclusion and future work is given.

2. Background and related work

2.1. Digital forensic frameworks for IoT and smart systems

Digital forensics is a collection of methodologies that have been produced and refined through the scientific community, tasked with the secure collection of traces from a crime scene, their examination, analysis and presentation of identified evidence which can help identify the malicious actors, their methods and motives, during an investigation of a security incident, often leading to legal action [28]. Over the years, and due to technological development, digital forensics has been refined into several subcategories, each specializing with incidents in different

environments, namely: cloud forensics, network forensics, IoT forensics, mobile forensics, memory forensics, data forensics [29].

For the purposes of this paper, we focus on network forensics. Network forensics mechanisms focus on security incidents that occur in networks, commonly using logs and captured packets to detect intruders and malicious acts. Experts who perform network forensic actions, make use of various tools for the correct collection and storage of evidence before analysis takes place. However, there is no single best process to follow during an investigation, giving rise to numerous digital forensic frameworks. In essence, a digital forensic framework dictates the appropriate steps to be taken by a professional during an investigation. It segments the investigation into distinct and autonomous phases that propose specific techniques and technologies for that phase. As previously mentioned in Section 1, most forensic frameworks proposed for a smart home crime scene focus primarily on the acquisition phase of the investigation. Multiple frameworks have been proposed [11,12,30]. However, no framework has been singled out as the preferred choice by professionals, since standardization is lacking, and varying circumstances require different approaches and tools [2,31,32].

Many researchers have developed forensic frameworks for the IoT [7,11–13]. For instance, Hossain et al. [12] and Hossain et al. [11], proposed Probe-IoT and FIF-IoT, respectively, two models which handle the acquisition of evidence from IoT devices in a forensically sound way for maintaining integrity and chain of custody, without violating the users' privacy. Meffret et al. [7] proposed a FSAIoT framework for the collection of state data from IoT devices. Following that, Cebe et al. [13] developed a Block4Forensic acquisition model designed for vehicular data collection. Both Probe-IoT, FIF-IoT and Block4Forensic base their implementations on the blockchain scheme. For Probe-IoT and FIF-IoT, a distributed public ledger was established, with multiple stakeholders maintaining copies of the produced ledger, while Block4Forensic used a fragmented ledger to reduce storage requirements. In all of the aforementioned cases, the information that is stored in the blockchain includes diagnostics about IoT devices and interaction between devices and other network entities. In contrast, FSAIoT uses centralized controllers in a local network to monitor and collect states and data transactions from the devices.

A significant portion of the literature regarding the development of new forensic frameworks for IoT and smart systems was based on the concept of distributed digital blockchains [12,12,30]. Hossain et al. [12] proposed Probe-IoT, Le et al. [30] BIFF and Hossain et al. [11] the FIF-IoT. These frameworks have a lot of similarities. To begin with, they are all designed around the concept of a distributed digital blockchain system being maintained by several relevant stakeholders, including but not limited to: Law enforcement, insurance companies and the manufacturer of IoT devices. Pre-defined roles dictate access rights for the stakeholders, while digital signatures ensure non-repudiation of events. To enable the use of digital signatures, some trusted entity plays the role of the Certification Authority, maintaining the public-keys that might be hacked by man-in-the-middle attacks [33].

Although the framework sometimes speeds up the investigation process, as the investigator would not need to physically attend the crime scene and collect data, it comes with some drawbacks. To begin with, they require either the introduction of dedicated devices that collect transaction data or the smart devices themselves to periodically transmit such data to on-line services that collect the transactions and incorporate them to the blockchain. This may cause increased charges either for new devices, or extra power consumption for the smart devices, with possible degradation of services due to constrained resources. Furthermore, these frameworks rely on the harmonious cooperation of multiple entities, while at the same time require (i) extra resources by Law enforcement to store the collected data, and (ii) trust by the IoT device owners that register their devices [11,12,30].

Babun et al. [34] developed a different acquisition model named IoTDots. Its functionality relies on first modifying the applications which control the smart devices so that logs and data can be re-directed to a remote database at runtime. A secondary module pre-processes and analyzes the gathered data to identify security incidents. The analysis is done by converting all collected data into bitmaps and through training a Markov Chain model on legitimate interactions. Although IoTDots considers collection, preservation and analysis phases of an investigation, it has some drawbacks. The framework requires modified applications to be used in order for IoTDots to function. Thus, users will be under constant surveillance with their activities being recorded, which may raise privacy concerns. Additionally, during the collection and transmission of the collected data, preserving the integrity of data is not considered. The data integrity and authenticity stage is vital and needs to be applied at the early stages of the digital investigation process as the data might be modified by malicious entities during either the collection, transmission or storage.

2.2. Deep learning for tracing and discovering threat behaviors

Traditional methods of network forensics include the use of fuzzy logic [35,36], Naïve Bayes classifiers [37], neural networks [36,37], support vector machines [38] . Deep Learning tends to be the preferred method of choice for the task of network forensics as, although training such models is time-consuming, their execution time is low and they identify more complex patterns, outperforming other choices especially when working with large volumes of data [2,14,17,39,40].

Deep Learning is a subsection of artificial neural networks, where the neural networks have a deep architecture that span multiple hidden layers [14,41]. Parsing logs, network traffic and documents demand a reverse-engineered code to identify indications of an attack, which is an iterative process that humans cannot easily perform. To that effect, multiple types of machine learning models have been used to harness their discriminative capabilities [2]. Recently though, deep learning has received more attention from the research community because it learns data and its variations in-depth through multiple generative or discriminative models. In deep learning, stacking tens/thousands of hidden layers together has been shown to increase the predictive capabilities of a neural network, allowing it to identify complex patterns in the data [14,18].

A common application of deep learning in the forensic and security research areas, is attack identification in network traffic. One such example by Shone et al. [14] is the development of an intrusion detection system based on a combination of a stacked non-symmetric deep autoencoder and a random forest classifier. Initially, two autoencoders are pre-trained, and their 'encoder 'parts are stacked, with the last one feeding its output to the random forest classifier. Evaluation of the KDD dataset depicted an accuracy of 89.22%. Azmoodeh et al. [40] proposed a deep learning approach, based on a convolutional neural network to detect 'Internet of Battlefield 'malware attacks. The network was trained on eigenvectors generated from the operation code sequence graph of the disassembled code of a mobile application. It was shown that this new method outperformed previous work in the field of malware detection, with results indicating accuracy and precision of about 98%.

Yuan et al. [42] proposed DeepDefence, a DDoS detection method based on deep learning. Their approach tested a combination of a convolutional neural network with three different

types of temporal-aware neural networks, namely the recurrent neural network (RNN), gated recurrent unit neural network (GRU) and long-short term memory neural network (LSTM) using the ISCX2012 dataset. Results indicated that both LSTM and GRU achieved the best performance at around 98%. Brun et al. [43] developed a dense random neural network method for the identification of attacks against IoT gateways. Two attack scenarios were considered, Denial-of-Service and Denial-of-Sleep which were represented in a custom dataset. However, accuracy of this method was reported to be comparable to a threshold detector.

Van et al. [44] built a NIDS based on Deep Belief Networks (DBN). To determine the best approach, the researchers compared the performance of a stacked Autoencoder (SA) with a stacked Restricted Boltzmann Machine (SRBM) on the KDD99 dataset. Results indicate that the SA displayed a smaller error in classification compared to the SRBM, although it required more time for training. Pektas et al. [45] proposed a deep MLP system to process network flow patterns and identify botnets, specifically the communications between C&C and bots. During pre-processing, graphs were generated from collected flows and grouped by communication endpoints which allowed them to generate new statistical features. The researchers concluded that deep learning presents acceptable accuracy for botnet identification in flow data, with the added bonus that feature selection is not necessary, as deep networks identify the best features.

Cheng et al. [46] developed D2PI, a system based on a Convolutional Neural Network (CNN) which classified collected traffic into either 'malicious 'or 'benign ', based solely on the payload. In order to train the CNN, the payloads of packets were extracted, their lengths adjusted to a predefined length and incorporated in a matrix. Results indicated that D2PI is a promising first step towards incorporating CNNs in deep packet inspection systems. Le et al. [47] produced a deep learning classification approach for the identification of different malware samples without the need for expert knowledge. The malware samples were converted to one-dimensional image representation and then used to train several different neural network models that combined convolutional layers which processed the input, with RNN and LSTM layers. Best accuracy was achieved through the CNN bi-directional LSTM, at 98.2% with a class re-balancing step.

Alrashdi et al. [48] developed an anomaly-based NIDS, called AD-IoT, that detects compromised IoT nodes, and based their design on Random Forest and Extra tree classifiers. Evaluation was performed on the UNSW-NB15, after reducing the feature size from the original 49 to 15. Results showed that the AD-IoT displayed a promising performance, although precision for detecting attacks was the lowest metric at 79%. Homayoun et al. [49] proposed BoTShark, a botnet detection and traffic analyzer based on Autoencoders and Convolutional Neural Networks (CNN). The researchers based their work on the ISCX dataset, utilizing netflow data for training and testing their method. Two versions of the BotShark were implemented, one based on stacked Autoencoders, where feature extraction is performed prior to classification and one based on a CNN, where each record was fed to the network individually. Results indicated an accuracy above 90%, with the autoencoder performing slightly better than the CNN, due to the reduced feature-set that it used.

Deep learning has been developed to detect and trace attacks from industrial IoT systems and their networks. De La Torre et al. [50] proposed a conceptual monitoring framework. They first surveyed the existing research on forensics, with emphasis on the many types of deep packet inspection (DPI). Reaching the conclusion that, relying on purely DPI solutions would be ineffective due to encrypted network traffic and the mutability of attack patterns, the researchers proposed a software-defined network-based (SDN) monitoring system. This conceptual system dictates the use of a forensic black-box, where monitoring traffic from smart grid networks and control stations would be gathered, eliminating difficulties with the acquisition phase. This black-box can then be accessed, and deep learning models applied for the identification of attack patterns, for both forensic and future prevention purposes.

As previously discussed, most of the existing studies have focused on acquisition approaches [11,12,30], or modifications to controller applications [34]. The new proposed framework, PDF is a viable alternative, as it harnesses network flow data which produce results without raising privacy concerns. Furthermore, it considers the analysis and examination phases which were overlooked by many frameworks that were presented, while not requiring the introduction of new entities or alterations to existing IoT and smart systems.

2.3. The particle swarm optimization algorithm

Particle Swarm Optimization (PSO) is an optimization swarm-based algorithm originally proposed in 1995 by Eberhart and Kennedy [51]. The algorithm identifies a solution by iteratively traversing the search space, and gaging its quality through the use of an objective function. The PSO algorithm is considered to be metaheuristic, since it does not rely on any assumptions about the underline problem, and is utilized in order to detect if not optimal, then "good enough" solutions in reasonable time [52].

PSO is often utilized to determine the value of a variable, as such, the search space that the PSO algorithm traverses, is defined as all the possible values that the variable can take. The PSO algorithm functions by spawning a population of particles, with each particle defined by their current position in the search space, the best position that the particle has observed so far, a global best position achieved by some particle in the swarm and a relative velocity [52,53]. The particles are initialized and set to traverse the search space randomly, with an objective function used to gage the wellness of the new position of the particle, and possibly update the local best and/or the swarm's best solution.

Since the introduction of the original PSO by Ederhart and Kennedy [51], the equation of which is given in Eqs. (3)–(6), researchers have proposed variants of the algorithm, that improve its performance for certain problems, or extend its usability. To begin with, altering the values of the learning factors θ_1 and θ_2 of Eq. (5), has a direct impact to the search pattern that the swarm focuses on as reported by Kennedy et al. [51]. For instance, by diminishing the global search (setting θ_2 very close to or equal to '0'), forces the swarm to emphasize the individual local search of each particle, while having $\theta_1 = \theta_2$ causes the swarm to gravitate towards the average of the global and local best solutions.

The standard PSO (SPSO) introduced by Shi and Eberheart [54] introduced an inertial coefficient (ω), which was multiplied to the particle's previous velocity in Eq. (5). By having relatively large values of inertia, the particles prioritize exploration of the search space, as their previous velocity has a greater impact to their new velocity in each step. The inertial weight can be initialized in any number of ways, such as randomly [55], it can be set as a positive non-zero value [54], or be a function of time either non-linear or linear [56,57] as given by the following gradually declining Equation:

$$\omega_t = \omega_{max} - \frac{i}{i_{max}} * \omega_{max} - \omega_{min} \tag{1}$$

Where ω_{max} and ω_{min} are pre-defined max and min weight values and i and i_max denote the current and maximum iterations of the swarm respectively. The reason for having a time-decaying inertia weight, is that it causes particles to explore the search space during early iterations, and then gravitate towards local search as inertia decays.

By reviewing the OPSO and SPSO, researchers observed that some particles' velocity tended to explode. In order to counter these issues, improvements were devised, such as introducing velocity clamping and constriction factors [58–60]. In order to manage the exploration–exploitation tradeoff, the velocity of each particle is calculated and limited to a pre-defined maximum value. By selecting large velocity maximums, particles explore greater areas of the search space, while smaller values focus the search to a limited area. The constriction factor was introduced as an alternative to the inertial coefficient. To apply the constriction factor, the OPSO equation for velocity (5) is adjusted, by multiplying the constriction factor K to the new velocity, with the factor given by the Equation:

$$K = 2/|4 - \phi - \sqrt{\phi^2 - 4\phi}|, \ \ where\ \phi = \theta_1 + \theta_2\ and\ \phi > 4. \quad (2)$$

Next, several PSO variants were developed in order to adjust the algorithm and enable its application in diverse problems [61]. The binary PSO [61] was developed, in order to enable the application of PSO to binary problems. Its novelty was to use the calculated velocity, produced by Eq. (5) with velocity constriction or clamping, as the input for a sigmoid function, and using the produced value (in the [0,1] range), to set the new position of the particle as either '0' or '1'. The fully informed PSO [62] is a variant of the SPSO, where a particle's movement is mostly affected by its neighbors. Specifically, instead of using the best position that the swarm has identified in order to update a particle's velocity, its neighbor's position is used. An altogether separate approach was taken with the cooperative PSO [63], where the problem was split, based on the dimension of the underline problem. In this variant, each swarm traverses a search space corresponding to a single dimension of the solution vector. As each swarm functions separately, special attention is given as to how the acquired partial solutions should be combined.

3. Overview of proposed particle deep framework

We present the stages of the new network forensic framework, so-called Particle Deep Framework (PDF), based on particle swarm optimization and deep learning for tracing attack origins and detect them from IoT networks, as depicted in Fig. 1.

The stages of the proposed framework are separately discussed as follows.

- **Stage 1: Network capturing Tools:**
  IoT devices have been placed into a network that is under investigation. The devices have been configured in a promiscuous mode, thus enabling them to see all traffic in a local network. Network packet captures are then carried out by utilizing network capturing tools such as Wireshark, Tcpdump and Ettercap. The collected pcap files are then forwarded to the data collection stage.
- **Stage 2: Data Collection and Management Tools:**
  This is the first stage in the network investigation process, where data are gathered in a form that can be further analyzed and examined, such as the datasets of BoT-IoT and UNSW-NB15. Initially, for the purposes of preservation, an SHA-256 hashing function [64] is employed to maintain the integrity of the collected data. Through this hashing function, the produced digests of the collected files can be used post-investigation to assert that the initial data have not been compromised. The collected pcaps are then processed by data flow extraction tools like Bro or Argus, that extract and summarize extract the network flows from the pcap files. An additional step during this stage is preprocessing, by handling missing and unuseful feature values, re-scaling and producing new features which can assist a model's training process. After filtering and cleaning datasets, the particle swarm optimization and deep learning models are applied to discover cyber-attacks and trace their origins.
- **Stage 3: Particle Swarm Optimization (PSO) for adapting hyperparameters of Deep Learning model:**
  The PSO algorithm [52] is chosen in order to adapt the hyperparameters of the deep model as it can easily address the local-optimum problem and quickly converges to obtain best fitness values compared with other evolutionary algorithms [53,65]. The PSO has been used to minimize/maximize an objective function, specifically in this study, it is used to maximize the Area Under Curve (AUC) values of a deep Multi-Layer Perceptron (MLP) algorithm to obtain the best hyperparameters that will be used to train and validate the algorithm for discovering cyber attacks and identifying their origins. To be more precise, this stage uses PSO to identify the optimal hyperparameters that maximize the AUC of the deep model.
- **Stage 4: MLP deep learning for attack identification:**
  The hyperparameters that have been estimated by the PSO algorithm and the collecting data from Stage 2 are used to train and test the MLP deep learning algorithm. The MLP was adopted by seven layers (excluding the input layer), with the number of neurons being: 20, 40, 60, 80, 40, 10, 1, as the best outputs were produced in terms of detection accuracy and false alarm rate. The three hyperparameters epochs, learning rate and batch size obtained from Stage 3 are used for training and validating the MLP algorithm. Data collection of stage 2 has been split into two groups: Training, and Testing split by 80%, 20% for measuring the performance of the MLP algorithm.
- **Stage 5: Performance measure:**
  Finally, performance metrics are obtained by running the trained deep MLP model on the Testing and validating data. Common performance metrics, including accuracy, precision, recall, false-positive rate, False-negative fate and f-measure, are estimated to measure discriminative capabilities of the proposed model. The details of Stages 3–5 are explained in the following two sections.

4. Particle swarm optimization (PSO) algorithm for deep learning parameter estimations

Particle Swarm Optimization is an evolutionary algorithm which was derived by observing swarms of fish and birds in nature [52]. A particle swarm is comprised of a pre-selected number of particles (P). During runtime, each particle is randomly initialized and starts propagating through the search space (v).

When a particle reaches a new position in the search space (v_{t+1}), an objective function is used to determine the quality of that position, with the function changing, depending on the problem that is optimized. Each particle is defined by a group of four vectors, its current position in the search space (x_t), its velocity (v_t), the best position identified by it (x_{lbest}) and the global best position (x_{gbest}), as declared in the following equations.

$$P = p_1, p_2, \ldots, p_n, n \in \mathbb{N} \quad (3)$$

$$\forall p_n \in P, p_n = (x_t, v_t, x_{lbest}, x_{gbest}) \quad (4)$$

$$v_{t+1} = v_t + \theta_1 * rand * (x_{lbest} - x_t) + \theta_2 * rand * (x_{gbest} - x_t),\ rand \in [0, 1] \quad (5)$$

$$x_{t+1} = x_t + v_{t+1} \quad (6)$$

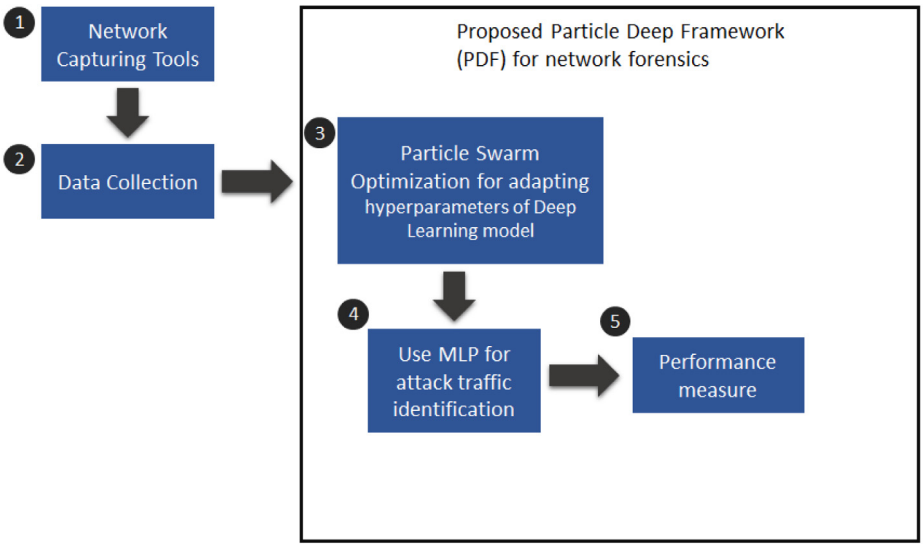

Fig. 1. Proposed network forensic framework using particle swarm optimization and Multi-layer Perceptron (MLP) deep learning algorithms.

In the above equations, v_{t+1} is the updated velocity of a particle. Thus the velocity of the particle at time t+1, depends on its previous velocity v_t, the learning factors θ_1 and θ_2 which are multiplied by a random number (rand) within the [0, 1] set and the distance between the local and global best solutions from the current position of the particle. The updated position x_{t+1} of the particle is calculated by adding the new velocity (v_{t+1}) to the previous position (x_t). In the following algorithm, a typical iteration of a maximizing PSO is given [52].

Algorithm 1: Particle Swarm Optimization maximization algorithm

P ← construct_particles(n_particles);
∀ p ∈ P, $p.X_{lbest} = p.x_0$, $p.X_{gbest} = -\infty$;
epochs ← load_epochs();
e←0;
while e < epochs **do**
  foreach p ∈ P **do**
    $v_{t+1} = v_t + \theta_1 * rand * (x_{lbest} - x_t) + \theta_2 * rand * (x_{gbest} - x_t)$, $rand \in [0, 1]$;
    $x_{t+1} = x_t + v_{t+1}$;
    if $x_{t+1} > x_{lbest}$ **then**
      $x_{lbest} = x_{t+1}$;
    end
    if $x_{t+1} > x_{gbest}$ **then**
      $x_{gbest} = x_{t+1}$;
    end
  end
end
return P.global_best()

Algorithm 1 is used to get the optimal hyperparameters of the MLP deep learning model to ensure the highest detection accuracy of detecting and tracing attack vectors. The particle swarm algorithm is initialized and tasked with maximizing the AUC values of the deep learning model, by seeking the best values of the hyperparameters: batch size, epochs, and learning rate, in their respective search spaces.

There are multiple reasons for using the PSO algorithm instead of another metaheuristic for the purpose of hyperparameter selection. To begin with, PSO is a simple algorithm to implement and its inner workings can be understood easily [66]. PSO does not guarantee an optimal solution however, it has been shown that it can produce satisfactory results in reasonable time [67]. Furthermore, although PSO tends to converge faster than other metaheuristic algorithms [68], it can be further sped up through parallelization of some of its parts, such as calculating the objective function's value. Finally, since our research has indicated that PSO has not been used for hyperparameter optimization, this work provides an indication as to how well it performs when applied for this task. In other words, this research provides empirical data about PSO performance, when tasked with optimizing deep MLP for application in the network forensic discipline.

5. Proposed particle deep model for network forensics

The proposed particle deep model is an important addition to the field of network forensics which spans the stages of network forensics, namely collection, preservation, examination/analysis and presentation, as shown in Fig. 2. The proposed model can be integrated to the investigation process of network forensics, by utilizing Deep Learning in the form of a neural network whose architecture is multi-layered, which in turn greatly improves its performance while maintaining a reasonable execution time.

Algorithm 2 presents the proposed particle deep model that integrates the PSO and MLP algorithms for improving the accuracy of attack detection and investigation, thus enhancing the computational process of the deep learning algorithm. The proposed model combines generative deep neural networks for the correct identification of malicious traffic, in a mixed environment comprised of both IoT and non-IoT traffic.

In order to determine the type of deep neural network to use in the PDF, the average performance of a vanilla deep MLP model [69] and Recurrent Neural Networks (RNN) with different number of steps [70] were compared. The goal of these tasks is to build a model that is as accurate as possible while maintaining a low false alarm rate.

In Algorithm 2 we depict a Particle Deep Model (PDM) iteration. First the neural network is loaded with its pre-selected layers and number of neurons. Initially, the three hyperparameters batch size, number of epochs and learning rate [b,e,lr] are randomly initialized. Next a particle swarm comprised of a pre-selected number of particles (*n_particles*) and number of iterations (*swarm_epochs*) is generated. Then algorithm 1 is utilized to identify the value of the hyperparameter that is being optimized, that maximizes the AUC value of the neural network

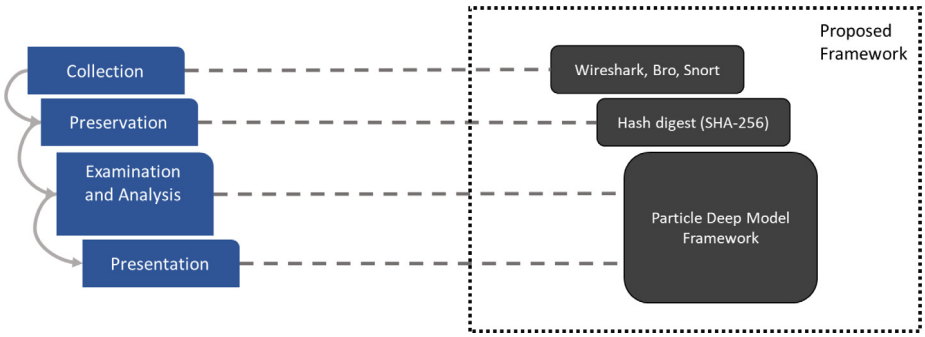

Fig. 2. Stages of investigation in network forensics including the proposed framework.

Algorithm 2: Particle deep model for hyperparameter estimation of deep learning

Data:
nn ← load_neural_network_structure() ;
[b,e,lr] ← initialize_random_hyperparameters() ;
hyperparameters ← [b,e,lr] ;
PS ← construct_particle_swarm(n_particles,swarm_epochs) ;
i ← 0 ;
foreach $h_1 \in hyperparameters$ **do**
  while $PS.swarm\_epochs \neq 0$ **do**
    $h_1 \leftarrow PS.maximize(nn.AUC, h_1) using algorithm 1$;
  end
  $nn.save\_opt\_hyperparam((h_1))$;
end
$nn.train\_NN(training\_set())$;

$(h_1 \leftarrow PS.maximize(nn.AUC, h_1))$. The process is repeated for every hyperparameter that is being optimized, the identified values of which are utilized to train the final neural network.

The Bot-IoT dataset [25] was utilized for training and testing of the deep models, with 80% of the dataset used for training while the remaining 20% for testing. On both the training and testing set, we performed min–max normalization, which resulted in the values being scaled data within the range of [0,1] to assert the neural networks models will not bias towards a particular class and ensure regularization of learning. We selected the type of deep learning model for our framework, and then we initially trained and validated it manually, through a trial-and-error process. After that, we employed the particle deep model, as explained in Algorithm 2, to optimize the hyperparameters of the deep neural network. The reasoning behind this action, is that there is no standard process for selecting the best values for hyperparameters, such as *the number of layers, the number of nodes for each layer, a learning rate*, during the pre-training phase of a neural network.

In the proposed particle deep model, the particles propagate through the searchspace of each of the three hyperparameters being optimized, one at a time. The logic behind individualizing the hyperparameter search was to make the search space smaller, since if the swarm was applied to all three hyperparameters at the same time, the search space would have been equal to *Batch_size_Size * Epochs_Size * Learning_rate_Size*, where *_Size* indicates all the possible values of the hyperparameter that can be estimated in their search space.

Hyperparameter values affect the training process of a neural network, and although there are many, we will focus on three, namely *epochs, batch size* and *learning rate*. The batch size determines the number of records (rows, if the data is in a structured form), which is parsed by the model before its weights are updated. The number of epochs indicates the times that the network will take the entire training dataset. The learning rate is a decimal, usually between (0,1), that is used to determine how much the weights are updated, with values close to 1 causing large updates that may be erratic and overshoot optima, while values close to 0 translates to very slow updates.

The logistic cost function was chosen, as it is suited for separating between attack and normal traffic, which is considered a binary classification problem [71]. Furthermore, due to class imbalance imposed by the nature of the attacks which have more records than normal traffic, weights were used to compensate for the imbalance. The logistic cost function is given in the following equation:

$$C = -\frac{1}{m}\sum_{i=1}^{m}(y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)) \tag{7}$$

Where m is the number of instances that have been fed to the model, y_i is the expected class feature value, and $\hat{y}_i$ is the calculated class feature value of the ith example.

The weights that were introduced for the two classes would be incorporated in the logistic loss equation like so: $w_1 y_i log(\hat{y}_i) + w_0(1 - y_i)log(1 - \hat{y}_i)$, with w_0 being the weight for the normal traffic and w_1 for the attack vector. Each particle trains a version of the deep MLP, with a different value for the trained hyperparameters, retaining their global best hyperparameters. The steps of training and validating data collection using the proposed particle deep model is discussed in Algorithm 3.

Algorithm 3: Steps for training and testing the proposed particle deep model

Data:
$S = 0 : batch, 1 : epochs, 2 : learning_rate$;
$Results = batch : -1, epochs : -1, learning_rate : -1$;
$NN = loadNN_structure()$#*Hyperparameters that arent trained*;
$n\_p = 6$#*number of particles*;
$n\_e = 4$#*number of epochs*;
$Results = randomInitialState()$;
$task =$ '*maximize_AUC*';
for $k = 0; k <= 2; k++$ **do**
  #*Runs once for each hyperparameter to optimize*;
  $particles = generateParticles(n\_p, n\_e)$;
  $bestHyper = runPSO(particles, task, S[k], NN)$;
  $Results[S[k]] = bestHyper$;
end
#*trains a model with the identified hyperparameters*;
$trainedNN = trainNN(NN, Results)$;
$testNN(trainedNN)$;

Because each particle needs to train an MLP to produce the AUC value, which is used in the search space to calculate speed

and position for the particle, the PDF runtime is excessive. For each hyperparameter, the PSO execution time was as follows: for batch optimization 4 h, for number of epoch optimization 3 h and for learning rate optimization 4 h, in total requiring 11 h for optimization. Additionally, training of the final deep MLP model required 7 min, while the prediction speed was 14,762 records/second. Regarding the time complexity of the proposed PDF, it is equal to $O(n_e * n_p * n_h) + O(mlp)$, where n_e is the number of epochs, n_p the number of particles that the PSO will use and n_h the number of hyperparameters to be optimized.

6. Architectural design of deploying proposed PDF in IoT networks

In this section, we discuss an architectural design that illustrates the deployment of the proposed particle deep framework in IoT networks of smart homes, as an example of current smart systems, as shown in Fig. 3. A typical IoT system architecture can be organized into three groups, the *IoT layer, network layer and the cloud layer* [2].

Our proposed framework could be easily deployed at the network layer, as its actions focus on tracing and discriminating between normal and attack vectors. Network traffic is usually encrypted on the Internet and real-world production networks, which hinders the analysis of payload data, while privacy laws may also cause problems. As such, network flow analysis is used in this research to train and validate the proposed framework to avoid Law enforcement and privacy restrictions.

In the IoT layer, smart devices operate and interact with one another, via a local communication protocol such as ZigBee, Bluetooth or WiFi. Collected data is transmitted and commands that are issued through the network layer, which typically involves a coordinator and a sensor bridge device on the IoT layer side. A protocol that is often used by IoT systems is Message Queue Telemetry Transport (MQTT). MQTT is an application layer protocol that sits on top of the TCP/IP stack and enables high-speed and low bandwidth network communication thus lowering power requirements for the devices involved [72]. It is worth mentioning here that MQTT is stacked under TCP, so attacks such as DoS, DDoS, scanning could be types of MQTT attacks.

The MQTT protocol is utilized, in order to publish and subscribe to network data that reach the Cloud layer, where users can access the collected data, or manage their IoT services. The Cloud layer is organized into four categories depending on the type of service they provide, which are Platform-as-a-Service (PaaS), Infrastructure-as-a-Service (IaaS), Software-as-a-Service (SaaS) and Thing-as-a-Service (TaaS) [73].

SaaS provides ready cloud-based software over the Internet, that is maintained by a third party and can be readily accessed by users. PaaS provides a platform, which comes with the tools necessary, including storage and cloud resources, for software development and maintenance. IaaS provides direct access to remote resources, either in virtual or in physical form, that the users operate and manage. TaaS is a newer concept, where network and data storage and analysis services are provided, in a way that makes them easy to integrate with deployed IoT systems. In this, TaaS resembles PaaS, as both provide configured platforms for development and analysis.

Essentially, a smart home is designed to offer comfort and efficiency through automation. As with IoT devices themselves, smart home deployments come in many shapes and forms. As such, they may be comprised of smart locks and music systems, which can be remotely activated and managed. For example, a realistic smart home deployment that was employed by Koroniotis et a. [25] to develop the Bot-IoT dataset, which has been utilized in this research, consisted of five IoT devices. Specifically, the IoT devices include a smart fridge, smart air-conditioning system, smart thermostat, smart lights and smart garage door. The proposed framework could be used to investigate security events involving Botnet activity and their origin in smart homes and other smart systems.

7. Experimental results and discussions

7.1. Datasets used and evaluation criteria

The Bot-IoT dataset [25] incorporates normal and attack traffic, including IoT traffic that was generated from Node-Red with 72.000.000 records and at 16.7 GB for the finalized dataset and combined pcap files at 69.3 GB. The UNSW-NB15 dataset [26] was generated by using the IXIA PerfectStorm tool, generating a number of diverse attacks, with the pcap files totaling at 100GB. We used these two datasets in this research, as they are both relatively new and both represent realistic normal traffic and attack scenarios. For training and testing, the Bot-IoT dataset was split in 80% and 20% respectively, resulting in 2.934.817 training and 733.705 testing records.

To evaluate and compare the trained models' performance, the following metrics are used:

- **Accuracy:** The fraction of correctly classified records from the total number of records.

$$(TP + TN)/(TP + TN + FP + FN) \tag{8}$$

- **Precision:** The fraction of records correctly classified as "Positive"from the records predicted as "Positive".

$$TP/(TP + FP) \tag{9}$$

- **Recall:** The fraction of records correctly classified as "Positive"from the total number of records that were "Positive".

$$TP/(TP + FN) \tag{10}$$

- **FPR:** The fraction of records falsely classified as "Positive" from the number of records that were "Negative".

$$FP/(FP + TN) \tag{11}$$

- **FNR:** The fraction of records falsely classified as "Negative"from the number of records that were "Positive".

$$FN/(FN + TP) \tag{12}$$

- **F-measure:** The harmonic mean of Precision and Recall.

$$2TP/(2TP + FP + FN) \tag{13}$$

The proposed particle deep framework was developed on a laptop outfitted with 16GB RAM, Intel Core i7-6700HQ CPU @2.6 GHz and an NVIDIA GeForce GTX 970M. Python code was used to build and train the DNN model, as-well-as identify the hyperparameters through PSO. Specifically, the following python packages were used: Numpy and Pandas for matrix manipulation and data pre-processing, Keras which provided a high-level interface with the TensorFlow backend package and Optunity for the PSO process [74].

7.2. Results and discussions

The experimentation results of evaluating the proposed Deep Particle Framework (DPF) using the evaluation metrics are explained in this subsection. In order to determine the neural network to use, a number of different architectures were tested, with their results presented in Table 1. The hyperparameters utilized are "epochs": 2, "batch_size": 512, "learning_rate": "0.001", with

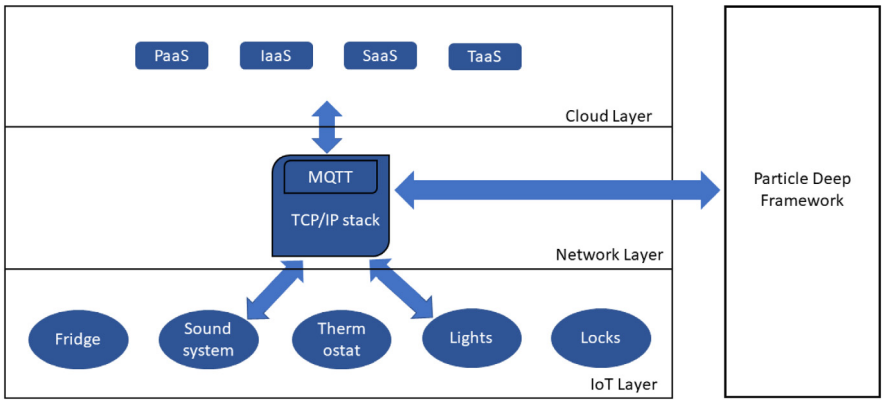

Fig. 3. Architectural design of deploying the proposed network forensic framework in IoT networks.

the activation function of hidden layers being "relu"and for the output layer "sigmoid". These hyper-parameters are by no means ideal for training but were arbitrarily chosen to help while comparing the effectiveness of the models. As such, we fed the output of the model to a single layer, single neuron Perceptron, with activation function "sigmoid", to obtain meaningful results.

Neural networks have been used to great success in a number of diverse fields, including cyber security and network forensics, proving their robustness and flexibility [75,76]. Among the various neural network versions, the Multi-Layer Perceptron (MLP) is considered to be simple but powerful as it is capable of modeling data that is non-linearly separable. Similarly, two neural networks that incorporate the concept of memory in their inner workings are the Recurrent Neural Network (RNN) and the Long–Short Term Memory RNN (LSTM-RNN), which have been applied in a significant portion of the work related to cyber security [77,78].

Recurrent Neural Networks for instance, have been employed in various Intrusion Detection System deployments [76,79]. Long-Short Term Memory RNNs improve on the RNNs by having the added ability to regulate when the network's memory should be updated, thus displaying better performance when classifying data in sequence. As this work seeks to identify the best neural network to use for the task of detecting Botnet activities in network captures where IoT devices were active, an empirical method was employed, where unoptimized neural networks were trained and tested on the same data. Then, by comparing their performance, the neural network best suited for the task of identifying botnet traces and differentiating them from normal traffic was identified. As such, the results of this empirical study are presented in Table 3.

An attempt was made to train an RNN with timesteps equal to Records, grouped by attack category, but handling this models output proved to be difficult. Additionally, the third, and deepest model could not be trained for timesteps=#records, as the memory requirements exceeded the existing RAM. As such, we grouped the Records in a window of 10 and trained the models (timesteps=10_Records).

Choosing timesteps=10 for the Bot-IoT dataset yielded the worst results. Even though it mostly displayed high values of accuracy (approximately 99.9% for most models), that value was misleading, as after considering the other metrics (specificity, recall, precision and fall-out) and viewing the confusion matrices, it became evident that the model could not identify the negative class (normal traffic).

It should be noted here, that in our experiments the models that were trained appear to achieve good performance, with high accuracy, precision and recall, with small values of FNR, but also indicate high values of FPR. This can be explained by the fact that the models that were tested were not optimized, and that the Bot-IoT dataset has an imbalance, with more records representing attack scenarios that normal network traffic. Thus, by misclassifying normal records as attack instances, the classifiers minimized the error value used during training. To avoid this issue while training the final three deep MLP models, which are compared in Table 4, weights were introduced to the logistic cost function.

In order to identify the best model to be used for the Network Forensic framework, Table 3 was constructed, which arranges the tested models in order, based on their average values, for the five metrics (Accuracy, Precision, Recall, Specificity and Negative Predictive Value), as given in Table 2.

As can be seen, the best performance on average was achieved by the RNN with 1 timestep model. Second, best was MLP and RNN with timeteps= features, while the worst performance was displayed when an RNN was trained with timesteps=10 (data appropriately re-shaped). It can be argued that an RNN with timestep=1 is equivalent to an MLP, with some added weight. As such, and based on the performances displayed here, the model best suited for the task of creating a network forensics framework appears to be the MLP model type.

A method for compressing the data was employed to investigate its effects on the performance metrics of the models. The process involves first passing each feature value through the Probability Density Function of the Normal Distribution, then multiplying the records (row-wise) with weights which were obtained by averaging the correlation coefficient matrix of the features and then adding the values, combining them into a single min–maxed normalized feature.

In order to tackle the class imbalance present in the Bot-IoT dataset between normal and attack traffic, we applied weights. Namely, the weights used were "1"for attack instances and "4500" for the normal instances. Following that, by employing the PDF, we obtained the following results for a deep neural network with the layers and neurons depicted in Table 4. Additionally, we explicitly set random seed values for both the PSO process and the neural network model, to ensure reproducibility.

In Table 4, we present three trained neural networks. Starting with (i), this is an initial, unoptimized neural network, for which hyperparameters were arbitrarily chosen. Although it achieved high accuracy, the model performed poorly, as it achieved a FPR of 0.8846 (88.46%). The model's accuracy is high, due to an imbalance between normal and attack traffic in the Bot-IoT dataset. To compensate for this, during training, we applied weights as discussed above.

Next, (ii) is an MLP where the initial 13 features were combined into one. We pursued this option, to reduce the already considerable training time. The compression process was previously described. It should be noted, that the accuracy of close

Table 1
Produced Metrics, organized by model structure and type.

Models\metrics			Accuracy	Precision	Recall	FPR	FNR	F-measure
Input: 13 (1*)	MLP	–	0.999	0.999	0.999	0.728	$9.913*10^{-6}$	0.999
		Timestep=1	0.999	0.999	0.999	0.796	$8.674*10^{-6}$	0.999
Hidden: 10, 4	RNN	(*)Timestep=Features	0.999	0.999	0.999	0.699	$4.956*10^{-6}$	0.999
Output: 1		Timestep=10_Records	0.999	0.999	1	1	0	0.999
Input: 13 (1*)	MLP	–	0.999	0.999	0.999	0.543	$3.717*10^{-6}$	0.999
		Timestep=1	0.999	0.999	0.999	0.388	$6.196*10^{-6}$	0.999
Hidden: 10, 8, 9, 6, 4	RNN	(*)Timestep=Features	0.999	0.999	1	1	0	0.999
Output: 1		Timestep=10_Records						
Input: 13 (1*)	MLP	–	0.999	0.999	0.999	0.223	$1.115*10^{-5}$	0.999
		Timestep=1	0.999	0.999	0.999	0.058	$3.841*10^{-5}$	0.999
Hidden: 20, 18, 10, 8, 4	RNN	(*)Timestep=Features	0.999	0.999	0.999	0.174	$1.858*10^{-5}$	0.999
Output: 1		Timestep=10_Records	0.999	0.999	0.999	0.917	$4.956*10^{-6}$	0.999
Input: 13 (1*)	MLP	–	0.999	0.999	0.999	0.359	$1.115*10^{-5}$	0.999
		Timestep=1	0.999	0.999	0.999	0.126	$1.858*10^{-5}$	0.999
Hidden: 40, 26, 18, 14, 10, 8, 6, 4, 2	RNN	(*)Timestep=Features	0.999	0.999	1	1	0	0.999
Output: 1		Timestep=10_Records	0.999	0.999	1	1	0	0.999
Input: 13 (1*)	MLP	–	0.999	0.999	0.999	0.233	$1.734*10^{-5}$	0.999
		Timestep=1	0.999	0.999	0.999	0.64	$1.487*10^{-5}$	0.999
Hidden: 8, 2	RNN	(*)Timestep=Features	0.999	0.999	0.999	0.203	$1.982*10^{-5}$	0.999
Output: 1		Timestep=10_Records	0.999	0.999	0.999	0.603	$1.115*10^{-5}$	0.999

Table 2
Average metrics.

Models \Metrics	Avg. Acc	Avg. Precision	Avg. Recall	Avg. FPR	Avg. FNR	Avg. F-measure
MLP	0.999	0.999	0.999	0.417	$1.065*10^{-5}$	0.999
RNN_Timestep=1	0.999	0.999	0.999	0.401	$1.734*10^{-5}$	0.999
RNN_Timestep=Features	0.999	0.999	0.999	0.615	$8.674*10^{-6}$	0.999
RNN_Timestep=10	0.999	0.999	0.999	0.903	$3.395*10^{-6}$	0.999

Table 3
Relative position of models, with regards to the metrics and their performance.

	1st	2nd	3^{rd}	4^{th}
Avg. Acc	MLP, RNN_Timestep = 1	RNN_Timestep = Features	RNN_Timestep = 10	–
Avg. Precision	MLP, RNN_Timestep = 1	RNN_Timestep = Features	RNN_Timestep = 10	–
Avg. Recall	RNN_Timestep = 10, RNN_Timestep = Features	MLP, RNN_Timestep = 1	–	–
Avg. FPR	RNN_Timestep = 1	MLP	RNN_Timestep = Features	RNN_Timestep = 10
Avg. FNR	RNN_Timestep = Features	RNN_Timestep = 10	RNN_Timestep = 1	MLP
Avg. F-measure	MLP	RNN_Timestep = 1	RNN_Timestep = Features	RNN_Timestep = 10

Table 4
Neural Networks that were trained.

	(i) Unoptimized NN	(ii) Optimized NN with compressed input	(iii) Optimized NN with 13-features input
Neurons per layer	13, 20, 40, 60, 80, 40, 10, 1	1, 20, 40, 60, 80, 40, 10, 1	13, 20, 40, 60, 80, 40, 10, 1
Epochs	2	12	12
Batch size	350	3064	732
Learning rate	0.2	0.0015	0.0015
Accuracy	0.999	0.947	0.999
Precision	0.999	0.999	1
Recall	0.999	0.947	0.999
FPR	0.884	0.081	0
FNR	$9.269*10^{-5}$	0.052	$9.541*10^{-5}$
F-measure	0.999	0.973	0.999

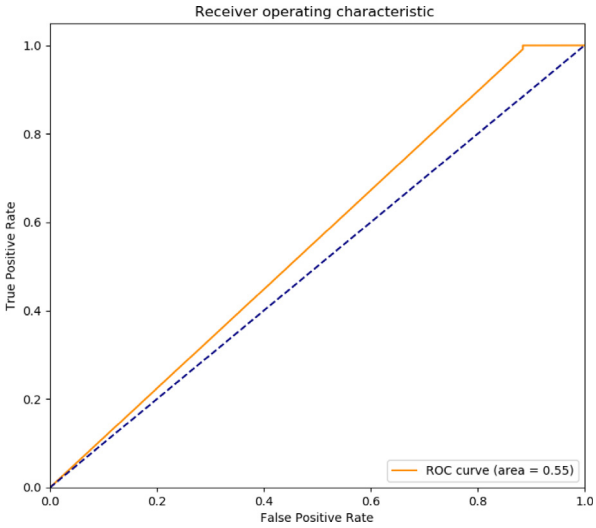

Fig. 4. (i) Unoptimized NN.

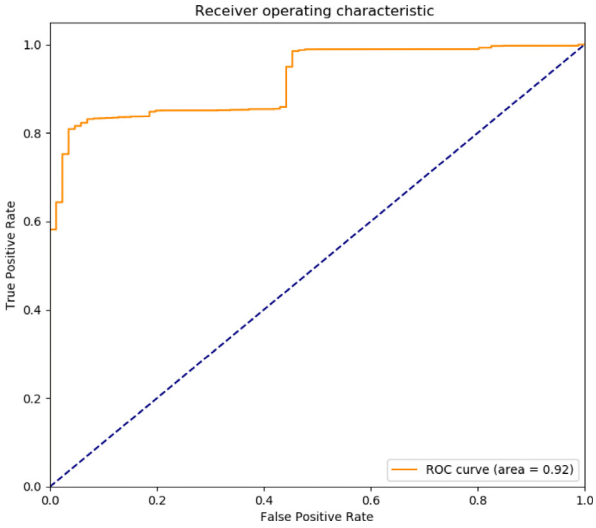

Fig. 5. (ii) Optimized NN with compressed input.

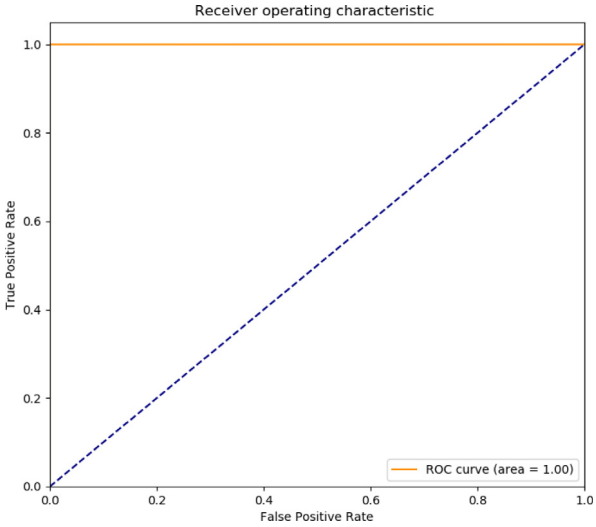

Fig. 6. (iii) Optimized NN with 13-features input.

to 95% was achieved, although FPR and FNR were not decreased further than 8% and 5% respectively. Finally, (iii) represents the optimized MLP with the original 13 features. Compared to compressed input NN (ii), the only difference in the trained hyperparameter values was the batch size, which for the 13-feature input NN (iii) was a smaller 732. This network displayed the best performance out of all the networks we trained, while also reducing the FPR and FNR to close to 0. The Receiver Operating Characteristic curves for these networks can be seen in (see Figs. 4–6).

There are a few reasons why the 13-feature NN (iii) outperformed the compressed input NN(ii). To begin with, the single input model compressed the available information from 13 features, quite possibly causing significant information loss. Additionally, the 13-feature MLP had more connections, and thus weights between the input layer and the first hidden layer. Specifically, (iii) had 260 trainable weights, while the compressed input NN (ii) had 20, which can lead the former to learn more complex patterns in the data. Fig. 7 depicts the 13-feature deep MLP (iii) model that was obtained by using the PDM.

From a network forensics point of view, the PDF is a useful tool. It can be used to detect attack traffic present in a packet capture file collected from a crime scene or a monitored network. By doing so, the origins of the attack can be established, assisting analysts with establishing a timeline of events, leading to the identification of the attack's target and sometimes the attacker's motive. Additionally, as it uses network flow information instead of performing deep packet inspection, privacy concerns are nullified. All of these concerns are important when introducing digital evidence in a court of Law.

The PDF has some advantages, for example, it automates the process of defining hyperparameters, arguably a difficult process as there exists no explicit way of defining a "good"set of hyperparameters. Furthermore, the used deep architecture makes the model capable of capturing complex patterns in data, improving its expressiveness. On the other hand, one disadvantage of the PDF is that it is time-consuming. Each particle needs to train a new version of the network on the newly discovered hyperparameter as it traverses the search-space. One way to combat this and reduce time is by parallelizing each generation, so as to make all the particles run at the same time.

8. Attack scenarios that validate the PDF

The dataset that was used to train the deep MLP model through the proposed PDF, was the Bot-IoT dataset [25]. As such, the attack scenarios that can be detected by the finalized deep classification model are presented here.

- Probing attacks [80] - are malicious information gathering activities, utilized by hackers in order to scan remote systems. Probing can be classified as either passive, where an attacker simply records traffic that is being exchanged by legitimate users and active, where an attacker exchanges traffic with the victim, gathering information about the latter's system through its responses. Depending on the purpose of the probe, these attacks are further split into Operating System and Service fingerprinting.
- Denial of Service [81] - are attacks that attempt to disrupt legitimate services, that are being accessed by users remotely. These attacks are split into two main subcategories, Distributed Denial of Service (DDoS) and Denial of Service (DoS) attacks, which are carried out by organized and compromised machines named Bots. As a secondary classification, DDoS and DoS attacks can be volumetric and protocol-based [82]. Volumetric attacks focus on flooding

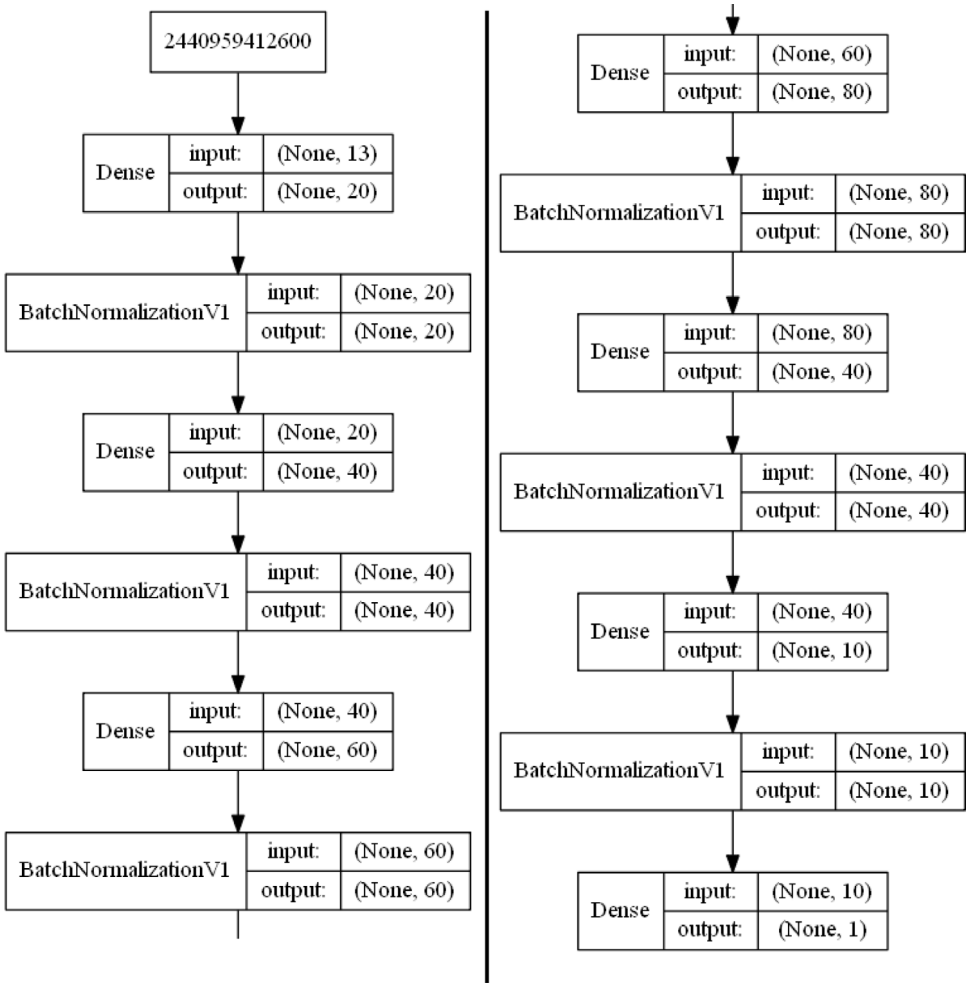

Fig. 7. 13-Feature Model structure.

their target with network traffic, while protocol-based attacks abuse weaknesses found in Internet protocols. The attack scenarios present in the Bot-IoT dataset include both DDoS and DoS with the following protocols used: TCP, UDP and HTTP.

- Information Theft [83]- is a category of attacks where the goal of the attacker is to extract sensitive data from the victim's machine. Based on the type of data that is targeted, these attacks can be split into two groups, data theft and keylogging. In the case of data theft, the attacker seeks to compromise the security of a remote machine and establish a reverse connection, through which data may be exfiltrated. On the other hand, in the case of keylogging, the attacker installs custom software on the remote machine, that records the keystrokes of the machine's user, effectively stealing any passwords that may be typed. The attacks in the dataset include both keylogging and data theft (exfiltration) scenarios.

9. Comparisons with other network forensics models-based machine learning

In this section, we compare the results obtained by the deep MLP that was trained by using the PDF, with results previously reported in other research studies. In order to evaluate the effectiveness of various machine learning models, [84] used the UNSW-NB15 dataset. The obtained results are compared to the new, optimized model. In addition, [25] trained some machine learning models to evaluate the Bot-IoT dataset, which was used in this research as well. These models include SVM, RNN and LSTM. As can be seen, the new optimized model improves on the performance of previous implementations.

As can be seen in Table 5, Table 6, the proposed deep optimized neural network (deep MLP), outperforms the other implementations that were trained both in the Bot-IoT and the UNSW-NB15 datasets respectively. The deep model achieved the highest accuracy and F-measure, while also the smallest FPR, FNR values, as the PSO can precisely identifies the best hyperparameters of the MLP and then the MLP can accurately discover cyber attack vectors and their attack families, as explained below.

One of the main reasons behind the high FPR and FNR displayed by the other ML models that were compared to the optimized deep MLP, is that the realistic cyber attacks that are represented in the data, are very close to the normal traffic. In other words, the DoS and DDoS volumetric attacks display characteristics similar to high-volume normal traffic, thus making discrimination between the two difficult. Furthermore, deep architectures such as the one used in this paper (deep MLP), that are comprised of multiple layers and neurons in each layer, render neural network capable of recognizing more complex patterns in the data.

The unoptimized Perceptron, Association Rule Mining (ARM) and Naïve Bayes classifiers produced highly erroneous results for various reasons. To begin with, the Perceptron lacked the necessary complexity, which is introduced through more layers and neurons, to model the patterns in the data. The ARM seeks

Table 5
Comparison of DNN with previous classifiers.

	13-Feature DNN model	SVM	RNN	LSTM
Accuracy	0.999	0.883	0.997	0.997
Precision	1	1	0.999	0.999
Recall	0.999	0.883	0.997	0.997
FPR	0	0	0.733	0.687
FNR	$9.541*10^{-5}$	0.116	$2.5*10^{-3}$	$2.492*10^{-3}$
F-measure	0.999	0.938	0.998	0.998

Table 6
Comparison of Bot-IoT generated DNN, with classifiers built on the UNSW-NB15 dataset.

	13-Feature DNN model	ARM	Decision tree	Naïve Bayes	Perceptron
Accuracy	0.999	0.856	0.932	0.727	0.639
Precision	1	0.908	0.948	0.92	0.642
Recall	0.999	0.895	0.944	0.627	0.984
FPR	0	0.255	0.09	0.095	0.97
FNR	$9.541*10^{-5}$	0.104	0.055	0.372	0.015
F-measure	0.999	0.902	0.946	0.746	0.777

to identify strong rules in the data, which may lead to some subtle patterns to be overlooked.

The Naïve Bayes, a probabilistic classifier, relies on the assumption of mutual independence of the features given the class, which often hinders classification. The Decision Tree's (DT) performance was the closes to the optimized deep MLP. One possibility for its performance, could be that DTs classify data by creating perpendicular lines to the axis/dimensions (features), and the data was mostly linearly separable.

Hyperparameter selection has tremendous impact on a neural network's performance. As selecting hyperparameters is not strictly defined by rigid rules that can guarantee best results, the process was automated in the PDF, with each hyperparameter tuned separately one after another. By tuning each hyperparameter separately, the dimensions of the search space that the swarm needs to traverse are reduced. As such, the swarm can run for fewer iterations and gradually build on previous results.

10. Identification of attack families and their statistics

In the Bot-IoT dataset, a number of botnet-related attacks are represented in the flows. These attacks form three groups, information gathering, information theft and Denial of Service (DoS) and Distributed DoS (DDoS). Information gathering attacks are activities which allow an attacker to identify the number, type and version of services of a remote machine such as service scanning and OS fingerprinting. Information theft attacks make use of previously exploited machines to steal sensitive information. From there, a bot can download documents present on the computer, or it can start recording keystrokes, thus stealing credentials, for example, keylogging and data theft attacks.

DoS attacks seek to render the remote services of a server unusable, and DDoS are multiple instances of DDoS to disrupt resources through protocols such as TCP and HTTP. Fig. 8 shows the detection rate of the attack types included in the Bot-IoT datasets. The proposed particle deep framework is capable of distinguishing between normal and attack traffic with an accuracy of about 99.9%. Prediction time for 733,705 records was 49.7 s, thus having a speed of prediction at 14,762 records/second. As such the model would be capable of identifying the attacks present in the dataset.

In the UNSW-NB15 dataset the network traffic that is represented includes both normal and a number of attack types. In total, nine attack types are represented in the dataset, which can be grouped into the following groups: Information gathering, Service disruption and Remote access. Analysis and Reconnaissance traffic belongs to the Information gathering group, as they seek to extract information from the remote target, ranging from open ports to identifying programs and their version that has run in the target.

The Service disruption group included Fuzzers and DoS attacks. Fuzzers are tasked with causing a program to suspend its normal operation by feeding it randomly generated traffic. DoS attacks rely on flooding the remote host with legitimate traffic, either depleting resources through sheer volume of information, or by exploiting known protocol bugs. The rest of the attack traffic types (excluding Generic and Worm) are categorized under Remote access. These attacks include Backdoors where, after using an Exploit, a Shellcode is delivered as a payload to the remote target and a channel is established that allows stealthy access to the remote target. Finally, Generic is a cryptographic-based attack that tackles block-cyphers with a given block and key size and Worms, which are viruses that replicate itself and propagates to other computes in a network. The proposed framework can identify and trace all the attack types included in the UNSW-NB15 datasets with an approximate 99.2% as depicted in Fig. 9.

11. Advantages and limitations of the PDF

The advantages of the proposed framework are the automation of hyperparameter selection, making manual optimization obsolete. Additionally, the collection stage utilizes software that is user-friendly and has been widely used in industry, thus it has credibility. Furthermore, integrity of data is considered and maintained through functions, something that is very important, as forensic investigations can be nullified if it cannot be ensured that the data has not been tampered. One disadvantage of the PDF, would be time of execution, as each move of a particle in each iteration requires time equal to the training time of the model we are optimizing. As such, the PDF's execution time is dependent to the volume of training data. A possible remedy to this issue, is the use of GPU and parallel computing for the training of different particles.

More relating to the underline data used to train the deep MLP, the simulated environment depicted several popular attacks, although it neglected to include any IoT-specific attacks, as the IoT devices were simulated through the use of Node-Red. Second limitation, is related to the amount of time necessary to tune and train a NN through PDF. As each particle propagates through the search space, it effectively trains a version of the NN, with a different value for the hyperparameter that is being tuned. Thus, depending on the size of the data, and the architecture of the NN, time may be excessive. Finally, the PDF in its current form utilizes network flow data. Although flow data bypasses some of

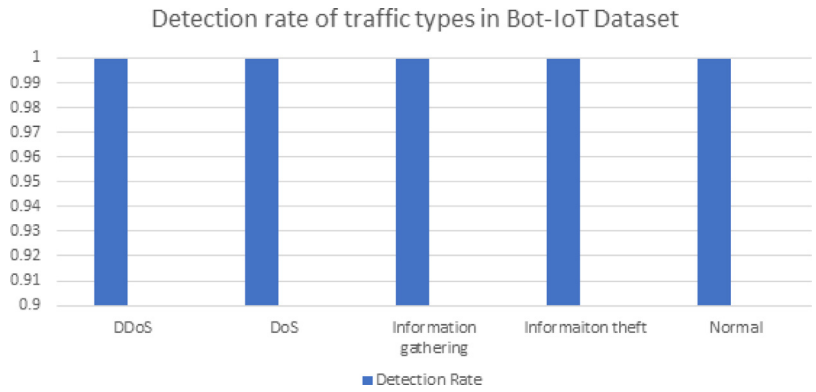

Fig. 8. Detection rate of attack types involved in the Bot-IoT dataset.

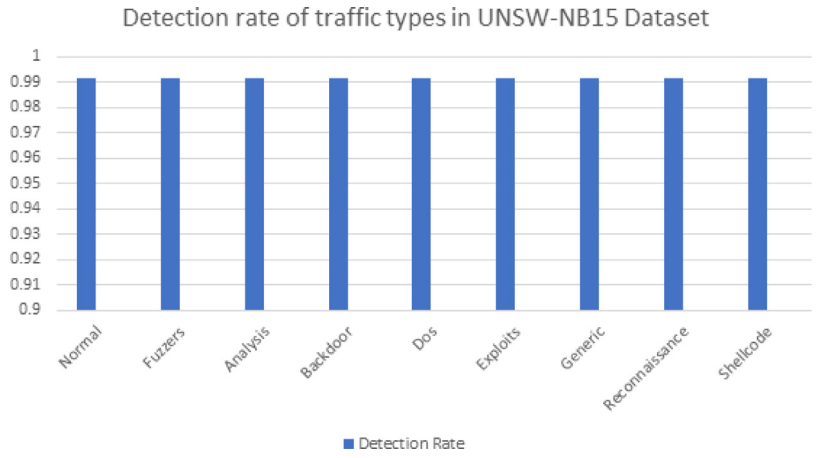

Fig. 9. Detection rate of attack types involved in the UNSW-NB15 dataset.

the restrictions brought about by DPI, it relies on traffic statistics, ignoring information such as the payload which could help in the identification of attack traffic. Furthermore, overcoming spoofing attacks, where the source IP has been altered, is a challenge.

Nevertheless, the proposed framework displayed a very high accuracy of detection. One reason behind the PDF's performance is the choice of a deep neural network, which is suited to handling large quantities of data by default. Another reason, is the utilization of PSO to identify the optimal hyperparameters that maximize the AUC of the model. Furthermore, compared to other frameworks [40,47] that focused on malware detection, the PDF bases its functionality on processing network flow data, the statistics of which can be easily adapted for use by a neural network.

12. Conclusion and future work

This paper has introduced a new network forensics framework that is named Particle Deep Framework (PDF) for discovering cyber-attacks and tracing them in IoT networks. This is because Security incidents that target IoT networks have been on the rise, as industry and the public adopt this new technology. The stages of the network investigation process were detailed and the proposed PDF was described. Particle Swarm Optimization (PSO) has been used to adapt the best hyperparameters of Deep Learning. Then, a Multi-layer Perceptron (MLP) neural network algorithm has been trained and validated using the Bot-IoT and network datasets for evaluating the performance of the proposed PDF. The proposed framework achieves high performances in terms of detection accuracy and timely processing compared with other machine learning models. Specifically, a deep MLP model was trained and tested on the Bot-IoT dataset, achieving an accuracy of 0.999, FPR of 0 and FNR of 9.5E−5 at a speed of 14,762 records per second. An architectural design that demonstrated the way of deploying the proposed framework in a network layer of IoT was described, using a smart home as an example of smart systems. In future, this work will be extended by applying the PSO to estimate multiple hyperparameters of various generative deep neural networks. Moreover, we will provide more insights for deploying in real-world IoT network in other smart systems such as smart health networks.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Nickolaos Koroniotis: Conceptualization, Methodology, Formal analysis, Writing - original draft, Writing - review & editing. **Nour Moustafa:** Methodology, Validation, Formal analysis, Writing - review & editing. **Elena Sitnikova:** Methodology, Resources, Writing - review & editing.

Acknowledgment

Nickolaos Koroniotis would like to thank the Commonwealth's support, which is provided to the aforementioned researcher in the form of an Australian Government Research Training Program Scholarship.

References

[1] S. Cook, 60+ IoT statistics and facts, Comparitech (2019) https://www.comparitech.com/internet-providers/iot-statistics/.

[2] N. Koroniotis, N. Moustafa, E. Sitnikova, Forensics and deep learning mechanisms for botnets in internet of things: A survey of challenges and solutions, IEEE Access 7 (2019) 61764–61785.

[3] B. Ali, A. Awad, Cyber and physical security vulnerability assessment for IoT-based smart homes, Sensors 18 (3) (2018) 817.

[4] S. Corporation, Internet Security Threat Report, Volume 24, Tech. Rep. 24, Symantec, 2019, URL https://www.symantec.com/content/dam/symantec/docs/reports/istr-24-2019-en.pdf.

[5] C. Robberts, J. Toft, Finding vulnerabilities in IoT devices: Ethical hacking of electronic locks, 2019.

[6] E. Ronen, A. Shamir, A.-O. Weingarten, C. O'Flynn, Iot goes nuclear: Creating a ZigBee chain reaction, in: 2017 IEEE Symposium on Security and Privacy (SP), IEEE, 2017, pp. 195–212.

[7] C. Meffert, D. Clark, I. Baggili, F. Breitinger, Forensic state acquisition from Internet of Things (FSAIoT): A general framework and practical approach for IoT forensics through IoT device state acquisition, in: Proceedings of the 12th International Conference on Availability, Reliability and Security, ACM, 2017, p. 56.

[8] S. Al-Sarawi, M. Anbar, K. Alieyan, M. Alzubaidi, Internet of Things (IoT) Communication protocols, in: 2017 8th International Conference on Information Technology (ICIT), IEEE, 2017, pp. 685–690.

[9] M. Conti, A. Dehghantanha, K. Franke, S. Watson, Internet of things security and forensics: Challenges and opportunities, Future Gener. Comput. Syst. 78 (2018) 544–546.

[10] R. Kaur, A. Kaur, Digital forensics, Int. J. Comput. Appl. 50 (5) (2012).

[11] M. Hossain, Y. Karim, R. Hasan, FIF-IoT: A forensic investigation framework for IoT using a public digital ledger, in: 2018 IEEE International Congress on Internet of Things (ICIOT), IEEE, 2018, pp. 33–40.

[12] M.M. Hossain, R. Hasan, S. Zawoad, Probe-IoT: A public digital ledger based forensic investigation framework for IoT, in: INFOCOM Workshops, 2018, pp. 1–2.

[13] M. Cebe, E. Erdin, K. Akkaya, H. Aksu, S. Uluagac, Block4forensic: An integrated lightweight blockchain framework for forensics applications of connected vehicles, IEEE Commun. Mag. 56 (10) (2018) 50–57, http://dx.doi.org/10.1109/MCOM.2018.1800137.

[14] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE Trans. Emerg. Top. Comput. Intell. 2 (1) (2018) 41–50.

[15] S. Prabakaran, S. Mitra, Survey of analysis of crime detection techniques using data mining and machine learning, J. Phys.: Conf. Ser. 1000 (1) (2018) 012046.

[16] Z. Wang, The applications of deep learning on traffic identification, BlackHat USA 24 (2015).

[17] G. Zhao, C. Zhang, L. Zheng, Intrusion detection using deep belief network and probabilistic neural network, in: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), vol. 1, IEEE, 2017, pp. 639–642.

[18] A. Zela, A. Klein, S. Falkner, F. Hutter, Towards automated deep learning: Efficient joint neural architecture and hyperparameter search, in: ICML 2018 AutoML Workshop, 2018.

[19] B. Wang, N.Z. Gong, Stealing hyperparameters in machine learning, in: 2018 IEEE Symposium on Security and Privacy (SP), IEEE, 2018, pp. 36–52.

[20] T. Chen, L. Zheng, E. Yan, Z. Jiang, T. Moreau, L. Ceze, C. Guestrin, A. Krishnamurthy, Learning to optimize tensor programs, in: Advances in Neural Information Processing Systems, 2018, pp. 3389–3400.

[21] J. Wang, J. Xu, X. Wang, Combination of hyperband and Bayesian optimization for hyperparameter optimization in deep learning, 2018, arXiv preprint arXiv:1801.01596.

[22] D. Stamoulis, E. Cai, D.-C. Juan, D. Marculescu, Hyperpower: Power-and memory-constrained hyper-parameter optimization for neural networks, in: 2018 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2018, pp. 19–24.

[23] S. Watson, A. Dehghantanha, Digital forensics: the missing piece of the internet of things promise, Comput. Fraud Secur. 2016 (6) (2016) 5–8.

[24] M. Chernyshev, S. Zeadally, Z. Baig, A. Woodward, Internet of things forensics: The need, process models, and open issues, IT Prof. 20 (3) (2018) 40–49.

[25] N. Koroniotis, N. Moustafa, E. Sitnikova, B. Turnbull, Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset, Future Gener. Comput. Syst. (2019).

[26] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-nb15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), 2015, pp. 1–6, http://dx.doi.org/10.1109/MilCIS.2015.7348942.

[27] N. Moustafa, J. Slay, UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-nb15 network data set), in: Military Communications and Information Systems Conference (MilCIS), 2015, IEEE, 2015, pp. 1–6.

[28] G.L. Palmer, A Road Map for Digital Forensics Research-Report from the First Digital Forensics Research Workshop (DFRWS)(Technical Report DTR-T001-01 Final), Air Force Research Laboratory, Rome Research Site, Utica, 2001, pp. 1–48.

[29] D. Joseph, J. Norman, An analysis of digital forensics in cyber security, in: First International Conference on Artificial Intelligence and Cognitive Computing, Springer, 2019, pp. 701–708.

[30] D.-P. Le, H. Meng, L. Su, S.L. Yeo, V. Thing, Biff: A blockchain-based IoT forensics framework with identity privacy, in: TENCON 2018-2018 IEEE Region 10 Conference, IEEE, 2018, pp. 2372–2377.

[31] A. Valjarevic, H.S. Venter, A comprehensive and harmonized digital forensic investigation process model, J. Forensic Sci. 60 (6) (2015) 1467–1483.

[32] L. Caviglione, S. Wendzel, W. Mazurczyk, The future of digital forensics: Challenges and the road ahead, IEEE Secur. Privacy 15 (6) (2017) 12–17.

[33] S.-W. Han, H. Kwon, C. Hahn, D. Koo, J. Hur, A survey on MITM and its countermeasures in the TLS handshake protocol, in: 2016 Eighth International Conference on Ubiquitous and Future Networks (ICUFN), IEEE, 2016, pp. 724–729.

[34] L. Babun, A.K. Sikder, A. Acar, A.S. Uluagac, IoTDots: A digital forensics framework for smart environments, 2018, arXiv preprint arXiv:1809.00745.

[35] N. Liao, S. Tian, T. Wang, Network forensics based on fuzzy logic and expert system, Comput. Commun. 32 (17) (2009) 1881–1892.

[36] A.A. Ahmed, M.F. Mohammed, SAIRF: A similarity approach for attack intention recognition using fuzzy min-max neural network, J. Comput. Sci. 25 (2018) 467–473.

[37] A. Yudhana, I. Riadi, F. Ridho, DDoS classification using neural network and Naïve Bayes methods for network forensics, Int. J. Adv. Comput. Sci. Appl. 9 (11) (2018) 177–183.

[38] K. Nguyen, D. Tran, W. Ma, D. Sharma, An approach to detect network attacks applied for network forensics, in: 2014 11th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), IEEE, 2014, pp. 655–660.

[39] K. Alrawashdeh, C. Purdy, Toward an online anomaly intrusion detection system based on deep learning, in: 2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2016, pp. 195–200.

[40] A. Azmoodeh, A. Dehghantanha, K.-K.R. Choo, Robust malware detection for internet of (battlefield) things devices using deep eigenspace learning, IEEE Trans. Sustain. Comput. 4 (1) (2018) 88–95.

[41] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (7553) (2015) 436.

[42] X. Yuan, C. Li, X. Li, Deepdefense: identifying DDoS attack via deep learning, in: 2017 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, 2017, pp. 1–8.

[43] O. Brun, Y. Yin, E. Gelenbe, Deep learning with dense random neural network for detecting attacks against IoT-connected home environments, Procedia Comput. Sci. 134 (2018) 458–463.

[44] N.T. Van, T.N. Thinh, L.T. Sach, An anomaly-based network intrusion detection system using deep learning, in: 2017 International Conference on System Science and Engineering (ICSSE), IEEE, 2017, pp. 210–214.

[45] A. Pektaş, T. Acarman, Botnet detection based on network flow summary and deep learning, Int. J. Netw. Manag. (2018) e2039.

[46] R. Cheng, G. Watson, D2PI: Identifying malware through deep packet inspection with deep learning, 2018.

[47] Q. Le, O. Boydell, B. Mac Namee, M. Scanlon, Deep learning at the shallow end: Malware classification for non-domain experts, Digit. Invest. 26 (2018) S118–S126.

[48] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, H. Ming, AD-IoT: anomaly detection of IoT cyberattacks in smart city using machine learning, in: 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), IEEE, 2019, pp. 0305–0310.

[49] S. Homayoun, M. Ahmadzadeh, S. Hashemi, A. Dehghantanha, R. Khayami, Botshark: A deep learning approach for botnet traffic detection, in: Cyber Threat Intelligence, Springer, 2018, pp. 137–153.

[50] G. De La Torre, P. Rad, K.-K.R. Choo, Implementation of deep packet inspection in smart grids and industrial Internet of Things: Challenges and opportunities, J. Netw. Comput. Appl. (2019).

[51] J. Kennedy, R. Eberhart, Particle swarm optimization (PSO), in: Proc. IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.

[52] D. Wang, D. Tan, L. Liu, Particle swarm optimization algorithm: an overview, Soft Comput. 22 (2) (2018) 387–408.

[53] F. Marini, B. Walczak, Particle swarm optimization (PSO). A tutorial, Chemometr. Intell. Lab. Syst. 149 (2015) 153–165.

[54] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: 1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No. 98TH8360), IEEE, 1998, pp. 69–73.
[55] R.C. Eberhart, Y. Shi, Tracking and optimizing dynamic systems with particle swarms, in: Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), vol. 1, IEEE, 2001, pp. 94–100.
[56] A. Nikabadi, M. Ebadzadeh, Particle swarm optimization algorithms with adaptive inertia weight: A survey of the state of the art and a novel method, IEEE J. Evol. Comput. (2008).
[57] K.E. Parsopoulos, Particle swarm methods, in: Handbook of Heuristics, Springer, 2016, pp. 1–47.
[58] R. Eberhart, P. Simpson, R. Dobbins, Computational Intelligence PC Tools, Academic Press Professional, Inc., 1996.
[59] M. Clerc, The swarm and the queen: towards a deterministic and adaptive particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 3, IEEE, 1999, pp. 1951–1957.
[60] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512), vol. 1, IEEE, 2000, pp. 84–88.
[61] J. Kennedy, R.C. Eberhart, A discrete binary version of the particle swarm algorithm, in: 1997 IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5, IEEE, 1997, pp. 4104–4108.
[62] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Trans. Evol. Comput. 8 (3) (2004) 204–210.
[63] F. Van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (3) (2004) 225–239.
[64] F. Armknecht, A. Dewald, Privacy-preserving email forensics, Digit. Invest. 14 (2015) S127–S136.
[65] X. Lu, M. Liu, A fuzzy logic controller tuned with PSO for delta robot trajectory control, in: IECON 2015-41st Annual Conference of the IEEE Industrial Electronics Society, IEEE, 2015, pp. 004345–004351.
[66] M.N. Ab Wahab, S. Nefti-Meziani, A. Atyabi, A comprehensive review of swarm optimization algorithms, PLoS One 10 (5) (2015) e0122827.
[67] M.R. Bonyadi, Z. Michalewicz, Particle Swarm Optimization for Single Objective Continuous Space Problems: a Review, MIT Press, 2017.
[68] M. Couceiro, P. Ghamisi, Particle swarm optimization, in: Fractional Order Darwinian Particle Swarm Optimization: Applications and Evaluation of an Evolutionary Algorithm, Springer International Publishing, Cham, 2016, pp. 1–10, http://dx.doi.org/10.1007/978-3-319-19635-0_1.
[69] L. Noriega, Multilayer perceptron tutorial, School of Computing. Staffordshire University, 2005.
[70] J. Schmidhuber, Deep learning in neural networks: An overview, Neural Netw. 61 (2015) 85–117.
[71] Z. Zhang, M. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: Advances in Neural Information Processing Systems, 2018, pp. 8778–8788.
[72] M. Houimli, L. Kahloul, S. Benaoun, Formal specification, verification and evaluation of the MQTT protocol in the internet of things, in: 2017 International Conference on Mathematics and Information Technology (ICMIT), IEEE, 2017, pp. 214–221.
[73] A. Celesti, D. Mulfari, M. Fazio, M. Villari, A. Puliafito, Exploring container virtualization in IoT clouds, in: 2016 IEEE International Conference on Smart Computing (SMARTCOMP), IEEE, 2016, pp. 1–6.
[74] M. Claesen, J. Simm, D. Popovic, B. Moor, Hyperparameter tuning in python using optunity, in: Proceedings of the International Workshop on Technical Computing for Machine Learning and Mathematical Engineering, vol. 1, 2014, p. 3.
[75] O.I. Abiodun, A. Jantan, A.E. Omolara, K.V. Dada, N.A. Mohamed, H. Arshad, State-of-the-art in artificial neural network applications: A survey, Heliyon 4 (11) (2018) e00938.
[76] M.S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, A.P. Sheth, Machine learning for Internet of Things data analysis: A survey, Digit. Commun. Netw. 4 (3) (2018) 161–175.
[77] H. HaddadPajouh, A. Dehghantanha, R. Khayami, K.-K.R. Choo, A deep recurrent neural network based approach for Internet of Things malware threat hunting, Future Gener. Comput. Syst. 85 (2018) 88–96.
[78] R. Chalapathy, S. Chawla, Deep learning for anomaly detection: A survey, 2019, arXiv preprint arXiv:1901.03407.
[79] C. Yin, Y. Zhu, J. Fei, X. He, A deep learning approach for intrusion detection using recurrent neural networks, Ieee Access 5 (2017) 21954–21961.
[80] N. Hoque, M.H. Bhuyan, R.C. Baishya, D.K. Bhattacharyya, J.K. Kalita, Network attacks: Taxonomy, tools and systems, J. Netw. Comput. Appl. 40 (2014) 307–324.
[81] S. Behal, K. Kumar, Detection of DDoS attacks and flash events using information theory metrics–An empirical investigation, Comput. Commun. 103 (2017) 18–28.
[82] S.T. Zargar, J. Joshi, D. Tipper, A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks, IEEE Commun. Surv. Tutor. 15 (4) (2013) 2046–2069.
[83] A. Jesudoss, N. Subramaniam, A survey on authentication attacks and countermeasures in a distributed environment, Indian J. Comput. Sci. Eng. IJCSE 5 (2014) 71–77.
[84] N. Koroniotis, N. Moustafa, E. Sitnikova, J. Slay, Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques, in: International Conference on Mobile Networks and Management, Springer, 2017, pp. 30–44.

Nickolaos Koroniotis is a PhD Candidate at the School of Engineering and Information Technology (SEIT), University of New South Wales (UNSW) Canberra, Australia. He was born in Athens, Greece in 1992. He received his B.S in Informatics and Telematics in 2014 and his M.S in Web Engineering and Applications in 2016 from the Harokopio University of Athens. He enrolled in UNSW Canberra to initiate his PhD studies in February 2017 in the field of Cyber Security with a particular interest in Network Forensics and the IoT.

Nour Moustafa is a Lecturer in Cybersecurity at the School of Engineering and Information Technology (SEIT), University of New South Wales (UNSW) Canberra, Australia. He received his PhD degree in the field of Cybersecurity from UNSW in 2017. He obtained his Bachelor and Master degrees of Information Systems in 2009 and 2014, respectively, from Helwan University, Egypt. His areas of interests include Cybersecurity, in particular, Network Security, Intrusion Detection, Threat Intelligence, Privacy-Preserving and Digital Forensics for Industry 4.0, Internet of Things (IoT), Cloud, and Fog Computing. His methodologies include Statistical Learning, Machine/Deep Learning, Big Data Analytics, and Artificial Intelligence (AI) Planning.

Elena Sitnikova is an academic and researcher and the Program Coordinator for the Master in Cyber Security program at the University of New South Wales (UNSW) Canberra. She holds an Honors degree in electrical engineering and a PhD in Communication Control systems. Her current research interests are in the critical infrastructure protection area, carrying out research projects in the area of intrusion detection (IDS) for control systems cyber security and Industrial IoT (IIoT). She is one of the first Australians to be certified in CSSLP — Certified Secure Software Lifecycle Professional. Elena is an award winning academic, holding a Senior Fellowship of the Higher Education Academy (SFHEA) and a national Australian Office for Learning and Teaching (OLT) Team Citation award for Outstanding Contributions to Student Learning.