A real time profiling method to detect attacks on IoT networks using LSTM and 1DCNN

Khalil, KB, Ben Kalboussi*

UMR 8201 – LAMIH, Polytechnic University of Hauts-de-France, Valenciennes, France, khalil.benkalboussi@uphf.fr

Farah, FB, Barika Ktata

Miracl Laboratory, ISSATSo, University of Sousse, Sousse, Tunisia, farah.barika.ktata@issatso.u-sousse.tn

Mourad, MA, Abed

UMR 8201 – LAMIH, Polytechnic University of Hauts-de-France, Valenciennes, France, mourad.abed@uphf.fr

Ikram, IA, Amous

Miracl Laboratory, Enet'Com, University of Sfax, Sfax, Tunisia, ikram.amous@enetcom.usf.tn

The use of Internet of Things (IoT) devices is progressively increasing in both industrial operations and everyday tasks. However, their distributed nature and limited resources make them particularly vulnerable to cyber threats, where compromising a single node can undermine the entire network. In this paper, we propose a real-time anomaly detection method that dynamically profiles typical network traffic patterns and applies deep learning techniques—specifically LSTM and 1D CNN models—to detect early signs of cyber-attacks in IoT environments. Any deviation from the established behavioural profile triggers further scrutiny, enabling rapid identification of both physical device manipulations and suspicious network transactions. The approach is evaluated using the Ton-IoT and Bot-IoT datasets, encompassing diverse normal and malicious traffic scenarios. Experimental results demonstrate that the proposed system achieves a 98% overall accuracy with a 5% false-negative rate, indicating its potential to improve security and resilience in IoT networks.

CCS CONCEPTS • Network monitoring • Security protocols • Machine learning • Sensor networks • Distributed architectures

Additional Keywords and Phrases: Network profiling, Real Time, Deep Learning, Intrusion detection system, Internet of things.

* Corresponding author

1 INTRODUCTION

The Internet of Things (IoT) has had a transformative impact across diverse application domains, including smart homes, agriculture, manufacturing, and healthcare. By interconnecting a growing number of heterogeneous devices, IoT systems enable real-time monitoring and actuation, thereby facilitating data-driven decision-making in critical environments. These decisions often pertain to energy management, transportation, medical care, logistics, and other sectors directly impacting human life and highly valued assets. Consequently, ensuring a high level of security in IoT systems is imperative.

However, the distributed nature and resource constraints of IoT networks make them particularly vulnerable to cyber threats. Compromising a single node can grant attackers access to the entire network, exposing organizations to severe breaches and theft of sensitive data. More sophisticated attacks, such as ransomware, can even disrupt military and medical equipment, putting lives and national security at risk [1]. Thus, early and accurate detection of malicious activity in IoT networks is a critical challenge.

Intrusion Detection Systems (IDS) are commonly employed to monitor networks and detect suspicious behaviour. Nevertheless, adversaries continuously evolve their strategies to evade signature-based detection, leading to an increase in IoT-based attacks reported annually. Compromised IoT devices can also be exploited to launch stealthy, distributed attacks on other networks [2]. Consequently, there is a growing need for IDS solutions that are not only precise and scalable but also adaptable to emerging zero-day threats. While numerous approaches have been proposed, most still struggle with scalability, detection accuracy, false alarm rates, energy consumption, and especially the identification of previously unknown attack vectors [3].

Many existing methods rely on signature-based detection, which, despite being accurate for known attacks, fail to identify novel threats for which no patterns or signatures exist. These approaches also demand substantial computational resources and frequent manual updates, making them unsuitable for real-time anomaly detection. Behavior-based or anomaly detection approaches have emerged to address these limitations, yet many current solutions lack the adaptability to distinguish between genuine and malicious network traffic, particularly when encountering new, unknown forms of malware.

In response to these challenges, we propose a novel profiling-based preprocessing method combined with a deep learning architecture that integrates one-dimensional Convolutional Neural Networks (1D-CNN) and Long Short-Term Memory (LSTM) models. First, our method profiles normal network behavior by extracting identifying features from network packets in a dynamic and time-aware manner. This establishes a reference profile that can adapt progressively as network conditions evolve. Subsequently, the CNN/LSTM model classifies incoming network traffic into suspected or genuine categories based on deviations from the established profile. This two-stage approach enables the rapid and automatic detection of both known and emerging threats.

Considering the heterogeneous nature of IoT systems, we adopt a similarly heterogeneous and flexible technology stack to optimize intrusion detection. Our main contributions are as follows:

- Lightweight IDS: We propose a two-stage detection technique that operates efficiently on minimal-performance platforms, making it suitable for resource-constrained IoT devices.
- Dynamic Profiling: Our profiling algorithm is both dynamic and time-aware, allowing the reference profile to adapt progressively and improve detection over time.
- Realistic Testing Environment: We test our approach under simulated real-time conditions that closely approximate real-world IoT deployments, ensuring practical relevance and applicability.

The rest of this paper is organized as follows: Section 2 discusses the state of the art in IoT traffic analysis and classification, and introduces raw traffic datasets commonly used in research. Section 3 details our proposed approach and methodology.

Section 4 presents our experimental results, performance analysis, and comparison with alternative solutions. Finally, Section 5 concludes the paper and suggests directions for future work.

2 STATE OF THE ART

In recent years, researchers have increased their interest in studying the performance of different IDSs in IoT systems on different levels. In order to evaluate our work a deep study is conducted on existent research studies.

2.1 Related work

We classify Intrusion detection systems into three categories:

2.1.1Machine Learning-based Intrusion Detection Systems

This part focuses on the use of machine learning techniques for intrusion detection in IoT systems. Many studies have explored the effectiveness of machine learning algorithms in detecting and mitigating intrusions in IoT networks. Some experimenters have proposed new machine learning-based approaches to enhance the accuracy and efficiency of intrusion detection systems.

[4] proposed a hybrid IDS for IoTs in the way of overcoming real network security challenges, by considering the combination of CNN along with LSTM networks. Their proposed model performs an outstanding performance in terms of having 99.52% classification accuracy, 98.97% F1-score with a very low false alarm of only 0.14% on the CICIDS2017 dataset. By combining the strengths of CNNs in extracting spatial features from network traffic with the LSTMs for capturing temporal dependencies, this hybrid design gives a strong solution for the detection of known and novel threats alike. The model is most notable for its scalability and real-time processing capabilities, thanks to which it is well suited to the dynamic and heterogeneous nature of IoT networks. Moreover, the low rate of false alarms reduces redundant notifications, further enhancing its feasibility in realistic applications.

While this model has these strengths, its high computational overhead is a huge limitation, mainly because the resource-constrained IoT devices have feeble processing power and small memory. That means further research on how to optimize the model for lightweight deployment with its high performance preserved is needed. The study showed, in general, the potential of deep learning in advancing IoT security, but it also revealed the need for striking a balance between accuracy and resource efficiency to make it implementable in practice.

[5] introduces a graph neural network framework for unsupervised node anomaly detection (NAD) using Individual Smoothing Patterns (ISP) and Neighborhood Smoothing Patterns (NSP). ISP captures node-specific smoothing divergence, while NSP quantifies neighborhood smoothness via Dirichlet energy. SmoothGNN integrates spectral GNNs, coefficients from NSP, and a novel anomaly metric (SMeasure). Evaluated on 9 real-world datasets, it outperforms 11 baselines (shallow, reconstruction, self-supervised models) by 14.66% AUC and 7.28% AP on average, with 75x speedup via precomputed propagation. Critically, scalability claims rely on precomputation, which may not generalize to dynamic graphs. Theoretical links between smoothing and spectral properties lack empirical validation beyond synthetic examples. Code is available, but real-world deployment challenges (e.g., hyperparameter sensitivity) are underexplored.

[6] adopts Extreme Learning Machine (ELM) in the effort to enhance the effectiveness Intrusion Detection Systems (IDS) for IoT which has relatively short development time and high degree of flexibility. Models employing ELM were evaluated on NSL-KDD and Distilled-Kitsune datasets demonstrating improvement in accuracy on binary and multi class classification with the model performing well for use of Kitsunes dataset in real time detection. However, there are some drawbacks including a relatively high false positive rate, inability to cope with nonlinear attack patterns and old data sets

such as NSL KDD, which do not solve modern problems. Although the potential for easy and fast intrusion detection is high, the work does not provide an outline of how the solutions can be practically implemented or assimilated to existing IoT systems.

The IDS proposed by [6] is based on the ResNet algorithm and reaches accuracy rate of up to 97.65%. The robustness of the model was highlighted by using three datasets for training and validation, including the popular NSL KDD and CICIDS2017 datasets. ResNet50, a 50-layer deep convolutional neural network, was used for intrusion detection and demonstrated its effectiveness in classifying network packets into normal and malicious groups.

[8] uses machine learning techniques for feature selection to improve the overall detection accuracy that reached 99.84 % using a hybrid CNN + LSTM model testing on UNSW-NB15 dataset. Yet the model seems to be unstable when testing on another dataset, X-IIoTID, that reached 93.21%.

[9] This work propose a deep learning-based IDS for the securing a smart IoT healthcare system. A hybrid CUDA-powered Long Short-Term Memory Deep Neural Network (cuLSTM-DNN) model is proposed to detect cyber threats. The research uses the CICDDoS2019 dataset to train and evaluate the model and compares performance with other hybrid classifiers like cuGRU-DNN and cuCNN-GRU. In corroboration, the results showed that the cuLSTM-DNN model exhibited greater accuracy (99.62%), precision, recall, and F-score. Furthermore, it offers the benefit of GPU for CUDA-accelerated processing, significantly mitigating the computational complexity and permitting testing in just 13.7 milliseconds. Yet, despite these advancements in research, some limitations are experienced. The model is founded on supervised learning accompanied by a strict labeling requirement, which may cause it to struggle when it comes to evolving patterns of attack. Besides, the computational burden of deploying deep learning models on resource-deprived IoT devices may be daunting.

[10] propose an architecture based on big data processing tools like Spark and Hadoop to efficiently process big amounts of network traffic and detect intrusions using Spark Mlib. The models tested are based on Random Forest, Decision Tree, Naive Bayes and Logistic Regression. The highest metrics reached are 99% F1, accuracy and precision.

[11] propose an architecture-based IDS, composed of a two-stage model to detect anomalies then to classify them according to attack types. The paper tests multiple deep learning models and reached an F1-score rate. Of 97.61%. This work has particularly our interest because it uses a technique of two stages detection. Although resource consumption is not indicated by the authors, the experimentation process focuses only on evaluating the performance in terms of detection rate and F1 score. Yet we assess that the M2-DAE feature extraction technique used by the authors is greedy in terms of resources that may not be available in restricted systems like IoT.

We adapt later in this paper a similar two stage architecture yet with much lighter profiling technique to detect anomalies. In the next part we study some examples of Signature-based intrusion detection systems

2.1.2Signature-based Intrusion Detection Systems

In [12] , IoT-FIDS is presented as a lightweight framework for anomaly detection systems tailored for resource constrained IoT environments. Besides service usage and packet headers, the system focuses on flow-based profiling which is communication pattern analysis of any standard device. This helps to avoid the complexity associated with normal machine learning models. Performance evaluations on the UNSW-NB15 and BoT-IoT clearly show that IoT-FIDS is very efficient with over 99% detection accuracy and in some attack scenarios, the F1 score was as high as 1.0, an indication that the system is highly efficient in recognizing and separating normal traffic flows from malicious ones. Additionally, when required to operate practically, the system can do this without incurring too many false positives. But there are some limits like dependence on predetermined communication profiles which could lead to inaccuracies especially in cases where a

new or different device is attached to the network. Additionally, IoT-FIDS is not optimized for handling encrypted communications, such as those based on the MQTT protocol, limiting its applicability in certain secure IoT setups.

[13] presented a machine learning model based on a combination of Isolation Forest and One-Class Support Vector Machine to detect anomalies. The authors demonstrated an extensive study on the impact of statistical data like the number of packets. The obtained results reached 94.37% AUC.

Network-based intrusion detection systems (NIDS) gather data on incoming and outgoing internet traffic in order to identify threats. These systems make use of both anomaly-based and signature-based detection techniques. While anomaly-based detection uses behavioral analysis to track occurrences against a baseline of "normal" network activity, signature-based detection includes comparing the gathered data packets against signature files that are known to be malicious.

[14] proposed a solution based on a hybrid deep learning model (CNN), an Anomaly-based IDS on IoT networks for the detection of various forms of attacks was developed. The used database is UNSW NB15. A result of respectively accuracy, precision, recall and F1 of 98.6%, 100%, 100%, 99% are reached.

[15] worked on the database UNSW-NB15 and also CICIDS2017. They proposed an approach for attack and anomaly detection based on SVM, decision tree, Random Forest and KNN for the defense and mitigation of IoT cybersecurity risks in a smart city. The metrics accuracy, precision, recall and F1 have reached respectively 95.45%, 95%, 95%, 95%. Although this type of IDS is commonly used and relatively effective, it presents limitations in terms of performance and resources. In restricted environments like IoT systems, resource consumption is a key feature that have to be considered. The previously mentioned type of IDS is proved greedy in terms of energy and resource consumption therefore does not meet the requirement of such limited systems.

In the next part we will discuss the Behavioral Profiling method for detecting intrusions.

2.1.3Behavioral profiling Intrusion detection systems

[16] introduces a federated learning framework enhanced with Long Short-Term Memory (LSTM) networks for identifying IoT device behaviours. The study utilizes periodic communication features extracted locally at each IoT node, preserving user privacy by avoiding raw data sharing. This approach addresses the complexity and diversity of IoT ecosystems, enabling decentralized modeling of device interactions. LSTM networks are employed to capture temporal dynamics, while feature selection through Chi-square tests and signal processing techniques like Discrete Fourier Transform (DFT) and wavelet analysis enhance the model's accuracy. Experiments on three datasets: UNSW IoT Traffic Dataset, Aalto Dataset, and SOSR-19 Dataset, demonstrate exceptional performance, achieving an accuracy of 99.9% across various scenarios. However, the framework faces limitations. It heavily depends on precise periodic feature extraction, which may not generalize well to IoT devices with non-periodic or irregular communication patterns. Furthermore, the reliance on computationally intensive LSTM models and federated learning infrastructures may pose challenges for devices with limited processing power or bandwidth. The study is also confined to benign traffic scenarios, limiting its applicability in adversarial contexts. Overall, while effective in its domain, the method's scalability and adaptability to broader IoT use cases warrant further exploration.

Some works use “manufacturer usage description” to be able to monitor and profile the behavior of Industrial IoT networks [17] yet it is not always the case in other IoT systems which characteristics are not as precisely defined, like Smart Home systems.

[18] has suggested a much more flexible method to profile the behaviour of IoT devices in a smart home following a more basic electronic approach. By measuring the power consumption of each device, the authors have been able to detect anomalies with 94.04% overall accuracy.

[19] proposed an anomaly-based intrusion detection solution based on profiling network transactions. Any deviation from the defined profile is considered an attack and is subject to further analysis. The solution is based on a MobileNet Convolutional neural network (MobileNetV3). The experimental results show that the proposed anomaly detection system gives results with an overall accuracy of 98.35%, precision of 99.31%, recall of 99.01%, F1 score of 99.16% and 0.98% of false-positive alarms.

[20] presented a two-tier Cyber-physical Systems detection and attribution methodology for attacks in industrial control systems (ICS). In combination with a fresh ensemble deep representation learning model, the Decision Tree is intended to detect attacks in an unbalanced ICS environment on the first level. In an IoT-enabled system, they developed a two-level ensemble attack detection based on a decision tree and unique deep representation learning. The suggested system is based on anomalies. The study used 17 feature ICS datasets. The final results reached an F1 score of 98%, precision of 98%, recall of 98%, and overall accuracy of 98.14%.

Table 1: State of the art

Author (Year)	Method/Approach	Datasets Used	Network	Real-time	Performance
Dong et al. (2025)	graph neural network	9 datasets	general networks	No	Speed X75
Ansar et al. (2024)	CNN + LSTM hybrid IDS	CICIDS2017	IoT	Yes	Accuracy: 99.52%, F1: 98.97%, False Alarm: 0.14%
Altamimi et al. (2024)	Extreme Learning Machine (ELM)	NSL-KDD, Distilled-Kitsune	IoT	Real-time on Kitsune dataset	High false positive rates
Shaikh et al. (2022)	ResNet50-based IDS	NSL-KDD, CICIDS2017, another dataset	general networks	No	Accuracy: up to 97.65%
Altunay et al. (2023)	Hybrid CNN + LSTM with feature selection	UNSW-NB15, X-IIoTID	IoT	No	Accuracy: 99.84% (UNSW-NB15), 93.21% (X-IIoTID)
Faten et al. (2023)	Big data tools (Spark, Hadoop) + ML (RF, DT, NB, LR)	Not explicitly stated (general traffic)	general networks	No	Accuracy, F1, Precision: ~99%
Giampaolo et al. (2020)	Two-stage architecture (M2-DAE feature extraction)	Not explicitly stated (general traffic)	general networks	No	F1: 97.61%
Mutambik et al. (2024)	IoT-FIDS (flow-based profiling)	UNSW-NB15, BoT-IoT	IoT	Yes	Detection accuracy: >99%, F1: up to 1.0
Giampaolo et al. (2023)	Isolation Forest + One-Class SVM	Not explicitly stated (general traffic)	general networks	No	AUC: 94.37%
Smys et al. (2020)	Hybrid Deep Learning (CNN)	UNSW-NB15	IoT	No	Accuracy: 98.6%, Precision: 100%, Recall: 100%, F1: 99%
Rashid et al. (2020)	SVM, Decision Tree, Random Forest, KNN	UNSW-NB15, CICIDS2017	IoT (Smart City)	No	Accuracy: 95.45%, Precision: 95%, Recall: 95%, F1: 95%
Du et al. (2024)	Federated Learning + LSTM	UNSW IoT Traffic, Aalto, SOSR-19	IoT	No	Accuracy: 99.9%
Krishnan et al. (2022)	Manufacturer Usage Description (MUD)-based profiling	Industrial IoT networks	IoT (IIoT)	No	-
Dilraj et al. (2019)	Power consumption-based profiling	Smart Home (custom setup)	IoT	yes	Accuracy: 94.04%

Author (Year)	Method/Approach	Datasets Used	Network	Real-time	Performance
Rose et al. (2021)	Anomaly-based IDS (MobileNetV3) using network profiling	Not specified (likely IoT scenario)	IoT	No	Accuracy: 98.35%, Precision: 99.31%, F1: 99.16%, False Positive ~0.98%
Jahromi et al. (2021)	Two-tier CPS detection + Ensemble Deep Representation & DT	17-feature ICS datasets	IoT	No	Accuracy: 98.14%, Precision: 98%, Recall: 98%, F1: 98%

It is very important to properly conduct experiments and tests suggested methods to successfully validate it. In order to do so, the choice of adequate datasets is a must. In this part we discuss some of the available datasets that are used in the literature review.

2.2 Datasets

KDD CUP'99 [21] was created in 1999 by removing features from the DARPA98 1998 dataset [22] , which replicates a LAN used by the U.S. Air Force. There are 41 features in KDD CUP'99, and there are four types of attacks: DoS, R2L, U2R, and probing. Many researchers continue to use the dataset despite its age. However, there are some disadvantages as well. First off, because the test set contains some attack that aren't included in the training set, the distribution of the records between the training and test sets is very different. Second, the training and test sets contain over 75% duplicate data, which can result in biased classification models. Most notably, the dataset does not contain SDN-specific properties and was not produced in an IoT environment.

In order to enhance the KDD CUP'99 dataset, NSL-KDD was developed by Mohi-ud-din [23]. The number of records was decreased by removing duplicates. Additionally, the classes were evenly distributed. However, this dataset does not accurately depict how current networks behave.

In 2015, Moustafa et al. [24] developed UNSW-NB15 dataset. The network traffic was produced using the IXIA tool. Two of the 49 features of UNSW-NB15, which are labels for binary and multi-class classification, are UNSW-NB15. The dataset includes both regular traffic and nine different forms of attack traffic, including fuzzing, backdoors, analysis, exploits, generic, worms, and shellcode. The dataset's biggest issue is that some attack types don't have enough samples.

Another dataset produced by the Canadian Institute of Cybersecurity is CICIDS2017 [25]. Their B-Profile system was used to generate realistic benign traffic. The dataset consists of both regular traffic and six different forms of attack traffic, including brute force, DoS, botnet, port scanning, infiltration, and web attacks.

ToN-IoT [26] includes a variety of heterogeneous data sources, including network traffic statistics, operating system datasets for Windows 7 and 10, Ubuntu 14 and 18, and IoT and Industrial IoT (IIoT) sensor telemetry datasets. The data was gathered from a large-scale testbed network that was realistically designed at the IoT Lab of UNSW Canberra Cyber, connecting numerous virtual machines, physical systems, hacking platforms, cloud, and fog platforms, and IoT sensors to mimic the complexity and scalability of IIoT and Industry 4.0 networks.

Bot-IoT [27] is a dataset that includes simulated IoT network traffic generated on a realistic testbed environment that allows for capturing complete network information, it presents a diversity of attack types that matches real world scenarios. The dataset is generated by recording the interaction of simulated IoT nodes on running on Ubuntu VMs and AWS IoT hub, along with a configuration of Mosquitto MQTT broker.

2.3 Discussion

In the three categories of IDS, we have sensed these main issues:

Although the majority of studied papers have demonstrated good results, almost none of them have discussed the importance of False Negative Rate metric, which represents the amount of non-detected attacks. This metric is simply mentioned in some works by calculating the F1 score which is proportional to Precision and Recall.

Another shortage in almost all the existent proposed models is the test and evaluation method. A dataset is pre-processed then split into training, validation and testing batches then used to train and test the models. Although this method gives promising results it is far from actual implementation and is just a static validation of an AI model. Such systems must be tested and validated in real time-like environments such as workbenches and simulators because the time factor and flow of data batches have huge impacts on the performance of the models.

The data preprocessing pipeline and features selection algorithm are often complicated algorithms that reduce the resource efficiency of the model, meaning that it is impossible to implement them on limited IoT systems.

For the databases that we have cited, many of them do not include IoT specific applications or protocols, yet many authors tend to use them to test approaches that are meant to detect intrusions in IoT environments.

Although the profiling method presented in some works is quite interesting, it presents some major issues. Most of the papers present a behavioural profiling method that is not dynamic and is dependent entirely on two parameters: 1) number of bytes flowing in and out of the systems. 2) the preset detection period: Hourly Profile, Daily Profile, and Weekly Profile. This configuration ignores the sudden changes in packet flow (if existed) and does not update predefined profiles based on the system normal usage evolution.

From the works discussed above we are interested in three aspects that we base our work on:

1) Double stage detection architecture:

This paradigm gives our model the ability to save resource consumption by a first stage of detection rather than a heavy one-time analysis.

2) Profiling engine

This method gives the ability to keep track of the behaviour of the system. We implemented a statistical method that is capable of detecting changes in behaviour with minimal resources.

3) Deep learning engine

In order able to accurately detect intrusions a deep learning algorithm is implemented, based on the literature, using a combination of one-dimensional CNN and LSTM classifiers.

3 PROPOSED APPROACH

The architecture of our proposed solution is based on two important components: 1) network profiling engine and 2) the deep learning classifier engine. The IoT gateway houses the network profiling engine, which is responsible for communication and data collecting. Given the increased processing demands, it may be possible to move this component to a different hardware platform while keeping the IoT gateway in close proximity (bridged with the gateway). This enables the component to profile the inter-device connections as well as the LAN network that the gateway is a part of.

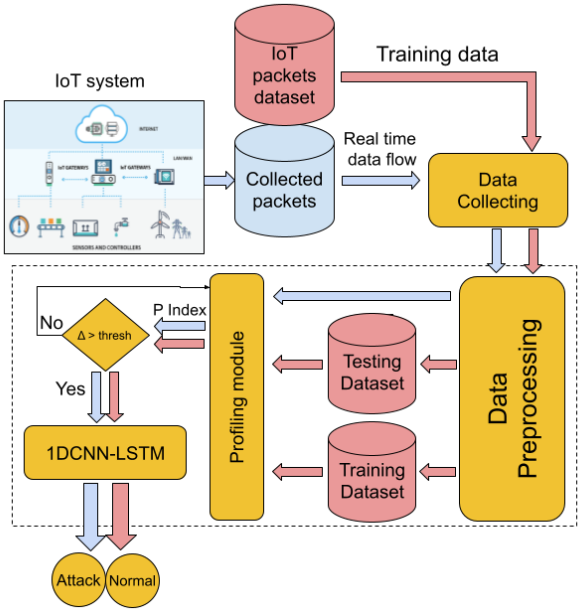

Figure 1: Architecture of the proposed model.

The fundamental benefit of the network profiling engine is the ability to calculate out-of-bound network profile behaviour. This behaviour is calculated by continuously monitoring the network traffic flow from each connected device (node). It uses rate-informed profiling to establish an anticipated pattern for each connected node on the system.

The proposed system operates within two phases as shown in Fig 1. First phase is called learning phase while the second is detection phase. In phase one a raw dataset is exploited, pre-processed, and used to calculate profiling properties. These properties are used to determine if a batch of packets could present a malicious threat. If yes, the packet features are injected in a one-dimensional convolutional neural network (1DCNN) combined with long short-term memory layers (LSTM) to give an effective deep learning model. In phase two the system is put in place and, unlike phase one, is functioning in real time. Captured packets are stocked in PCAP files which are pre-processed following a pre-set time interval. The profiling properties are then calculated in real time and according to it, packets are sent to the DL model for classification. The output of the classifier is reconsidered in the profiling engine, so the profiler keeps track of malicious traffic in a way that previous abnormal traffic is directly flagged in the corresponding profile and easily recognized.

The network profiling engine is closely connected to the IoT gateway. This enables the engine to access and profile both the inter-device connections and the LAN network that the gateway is a part of. After the engine scans of visible nodes, each device is profiled utilizing information related to network including routing information, hostname, temporal stamp, packet size, number of packets, etc. The effectiveness of this method is not related to the nature of the network itself.

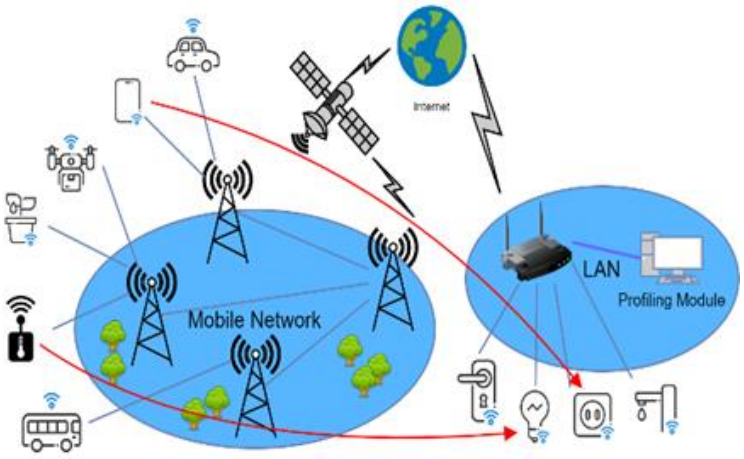

Figure 2: Integration of the Profiling engine.

As demonstrated in Fig 2, the IoT network layer is often composed of different technologies: Mobile (GSM), LAN (WIFI), LoRa, Sigfox etc. In our system, the profiling engine is integrated in the local area of the network, directly connected to the LAN gateway, a command sent from a cellular phone connected to a mobile network, to a power plug connected to a LAN (represented in Fig 2) is successfully detected by the profiling engine because the gateway represents an essential checkpoint for all incoming and outcoming traffic. Yet a packet sent from the same mobile phone to an irrigation system connected on the same mobile network is undetectable by our proposed system as it does not interact with the local network (in the case of serverless systems). Such packets are detectable on the application layer of the system.

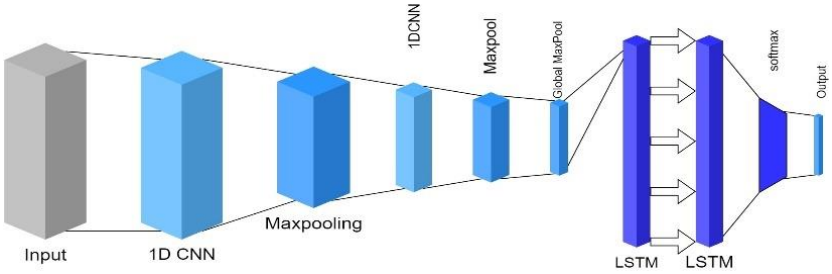

Figure 3: CNN-LSTM classifier model.

3.1 Profiling engine

In order to successfully profile each connection for each node a complex algorithm is put in place. The proposed method is based on vectorizing the statistical and temporal characteristics of connection flow to calculate a vector containing three information 1) Origin 2) Direction and 3) Length. Each of these characteristics will later be used to calculate the Profile Index (PI), which is considered as a session-like index. After the pre-processing stage, destination and source IP are used to calculate the origin. Direction is calculated based on the communication protocol and the length of the arrow is represents the number of packets in a pre-set time period. An example vector is demonstrated as follows:

$$V_n = (dsIP = \{dstIP, srcIP\} \quad , \quad Pr \quad , \quad Np = \sum_{i=0}^{t} p) \qquad (1)$$

Where dsIP equals {the destination IP, source IP} after converting the IPs to numerical form and scaling to a range of small number. It is used as coordinates in a two-dimensional plane. Pr represents the direction, and it is the protocol used in the traffic. It is a way of regrouping a node communication packets by protocol. Pr is calculated after converting the protocol into categorical form then scaling. Np is calculated by counting the packets circulating in the dsIP channel and on the Pr protocol during a pre-set time range t

After the extraction of profile vectors for each node, a matrix is created for each matching dsIP. the Profile Index is obtained by calculating the product of matrix for each node as demonstrated in the following equation:

$$PI = \sum_{i=1}^{n-1} V_n \,.\, V_{n+1} \qquad (2)$$

After calculating the Profile Index, the profiler periodically collects the network traffic from the gateway's LAN, which is then processed by the same profiling mechanism and using the same t value to produce a new PI measure. Comparing the PI of the new capture to the old value, a percentage difference is calculated:

$$\Delta = \frac{nPI}{oPI} \times 100 \qquad (3)$$

Where Delta (Δ) is the ratio of the calculated rate metric nPI, to the old rate metric oPI. The node's network activity is identified as out of profile and a classifier of the network traffic starts to look for any potential attacks if the delta value exceeds a threshold value that can be customized depending on network volatility. Even on a small network, this process won't negatively affect network performance due to its minimum network impact because the profiling period (t) can be changed in accordance with performance. Depending on how volatile the network is, this threshold can be increased or decreased. In the next section we will explain the different components of the proposed classification algorithm.

The used profiling method depends on the actual and the previous PI, in a way to calculate the profile difference between two consecutive time periods. A traditional way to profile network traffic is to calculate the profile image at first and save it then calculate the actual profile and compare it to the saved image. These methods do not take in consideration the temporal aspects of network usage and is sensitive to fluctuations in normal usage intensity between different time periods, as a result it could falsely detect anomalies if the profile image and the actual profile are different due to unique event usage. To overcome this limitation, we have opted to integrate a novel dynamic profiling method. Because the calculated Delta (Δ) represents the difference between two consecutive time periods, the minor changes in the system usage between two periods is not considered an anomaly, yet as you go through time the actual calculated profile could be very different from the early calculated profiles. This is implemented to compensate the possibility of system usage difference between two extended time periods which directly impacts the usage density. E.g. the usage of an IoT parking system in the morning (beginning of working time) and during working time.

ALGORITHM 1: Profiler algorithm

```
if (check data_file exist) then
    Read data_file
    Time ← 0
    Np ← 0
    t ← 20
    DThresh ← 10
    oPI ← 0
    V ←[]
    while true do
        while time < t do
```

```
            time++
            for line in data_file do
                if (line.Timestamp == time) then
                    if ( [ {line.dstIP, line.srcIP}, line.protocol ] exist in V ) then
                        V.dsIP ← {line.dstIP, line.srcIP}
                        V.Pr ← line.protocol
                        V.Np ← Np+1
                    End if
                End if
            End for
        end while
        V ← V.groupBy(dsIP)
        for line in V do
            PI ← PI x V
        end for
        Delta ← PI/oPI x 100
        If Delta > Dthresh then
            AlertAnomaly
        end if
        oPI ← PI
end while
```

3.2 Classification engine

In our proposed classification engine, the Complementary Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM) networks are integrated in order to correctly process and analyze the raw IoT packets. Feature extraction is the responsibility of the Convolutional Neural Networks (CNN) which are able to detect spatial patterns and hierarchies in raw packet data. This is performed by convolutional layers that learn the specific local features such as byte-level relationships and dependencies allowing the model to learn relevant traits that could differentiate benign and malicious traffic. In contrast, LSTMs have an advantage processing information through time therefore, they are able to comprehend the sequence of events that network traffic undergoes. IoT traffic could be characterized by time-sensitive features such as peaks or drops in behaviour patterns over time which LSTM could predict well. As such, the CNN-LSTM model combines the two in a way that features CNN's spatial conforming characteristics and LSTM's temporal features to provide a comprehensive structure for detecting attacks and anomalies in IoT devices. This combination helps to ensure high levels of accuracy are maintained during detection regardless of the complexity or evolution of the attack patterns.

Based on the output of the profiling engine, the Classifier engine is executed on raw packets captured. This section introduces the traffic classifier we put within our work, which learns and categorizes traffic packets by combining CNN and LSTM. Fig 3 demonstrates the classifier's general architecture. The model is composed of two parts. First part is composed of an input fully connected layer, a convolution layer, the pooling layer, a second convolution layer, another pooling layer, and an output global pooling layer. The CNN section analyses a pre-processed file after receiving it and

sends a high-dimensional package vector to the LSTM section. The first and second LSTM layers, fully connected layer and output layer make up the LSTM section. It is capable of processing a number of high-dimensional package vectors and producing a vector that shows the likelihood that each class the packet belongs to. The classification's final outcome is output by the SoftMax layer using a probability vector. LSTM is a specialized RNN created to address gradient disappearance and gradient explosion issues. It is powerful when performing extended sequence training. LSTM networks achieve better processing timing prediction thanks to their distinctive forgetting and selective memory gates. To save training time, the initial LSTM layer features a linear activation function. Through the ReLU function, the second LSTM layer is nonlinearly triggered.

After each CNN layer a Maxpooling function is used. It primarily serves to speed up and improve the efficiency of following computations by reducing the spatial dimensions of the feature maps. Maxpooling keeps the most relevant features while tossing out the less crucial aspects, which also makes the model more resilient to minor fluctuations in the input data. Overall, Maxpooling is used to reduce computation time and resources.

It is important to explore and analyse data before the learning process. In the next chapter we analyse the used datasets to train our model.

3.3 Dataset analysis

As we have chosen to test our model on both ToN-IoT and Bot-IoT datasets, we will further investigate its characteristics. Below is a detailed description of attacks contained in these datasets.

A threat model has been utilized as described by Alsaedi et al. [26] to simulate realistic attacks for the ToN-IoT dataset. These attacks are then monitored, and observations are labelled as either normal or attack, and which type of attack is performed. The following nine attack families were utilized in the datasets:

1) Scanning attack

Nessus and the Kali Linux Nmap tools were executed against the target subnet 192.168.1.0/24 and all other public vulnerable systems such as the Public MQTT broker and the vulnerable PHP website.

2) Denial of Service (DoS) attack

DoS attack scenarios, created with Python scripts using the Scapy package, were utilized against any vulnerable element in the IoT testbed network.

3) Distributed Denial of Service (DDoS) attack

DDoS attacks were executed against several nodes, using Python scripts and the ufonet toolkit.

4) Ransomware attack

Ransomware attacks were executed against targets running Windows OS that are monitoring IoT services, by means of Metasploit exploiting Server Message Block vulnerabilities.

5) Backdoor attack

The offensive systems with IP addresses 192.168.1.33,37 keep persistence in compromised machines by Metasploit executing a bash script with the command ”runpersistence-h”.

6) Injection attack

Various offensive systems inject data to web applications of Damn Vulnerable Web Application (DVWA) and Security Shepherd VMs, and to webpages of other IoT services. These attacks include SQL injection, client-side injection, broken authentication and data management, and unintended data leakage.

7) Cross-site Scripting (XSS) attack

XSS code was injected to the same machines and services as targeted with Injection attacks, by means of malicious bash scripts of Python code using the XSSer toolkit.

8) Password attack

Kali Linux Hydra and Cewl toolkits were used. They were configured by means of automated bash script to concurrently launch password hacking attacks against any vulnerable nodes in the testbed.

9) Man-In-The-Middle (MITM) attack

Kali Linux Ettercap was employed to execute ARP spoofing, ICMP redirection, port stealing and DHCP spoofing attacks.

The majority of the previously mentioned attacks are strongly related to temporal and statistical characteristics like the duration of connection, number of packets, packet size and the frequency of communication. These are considered important features in profiling communication traffic therefore are calculated and produced as output of the first part of our solution is discussed later in the proposed approach section. Yet to further comprehend the composition of the used dataset, an overview of the most important data features is shown in table 2.

After an extensive examination we noticed that packet samples in both datasets are extremely unbalanced. After joining the .CSV files from Ton-IoT, the result dataset contains a total of 21978630 samples, where normal samples are 788599 representing almost 3.6% of samples and the rest are divided between the attack types. The distribution of data between classes is very inconsistent, which can surely cause overfitting problems in deep learning models. After analyzing Bot-IoT we discovered the same problem, the same process is applied to it to be able to use it in our tests.

In the next section we explain the algorithms we setup for processing the data and applying the deep learning model in combination with explained profiling component.

Table 2: Features description

ID	Features		Description
	Ton-IoT	Bot-IoT	
1	ts	Stime	Timestamp of connection between flow identifiers
2	Src_ip	Saddr	Source IP addresses which originate endpoints' IP addresses
3	Src_port	Sport	Source port which originate endpoint's TCP/UDP ports
4	Dst_ip	Daddr	Destination IP addresses which respond to endpoint's IP addresses
5	Dst_port	Dport	Destination ports which respond to endpoint's TCP/UDP ports
6	proto	Proto	Transport layer protocols of flow connections
7	service	-	Dynamically detected protocols, such as DNS, HTTP and SSL
8	duration	Dur	The time of the packet connections, which is estimated by subtracting 'time of last packet seen' and 'time of first packet seen'
9	Src_bytes	Sbytes	Source bytes which are originated payload bytes of TCP sequence numbers
10	Dst_bytes	Dbytes	Destination bytes which are responded payload bytes from TCP sequence numbers

ID	Features		Description
	Ton-IoT	Bot-IoT	
11	Conn_state	State	Various connection states, such as S0 (connection without replay), S1 (connection established), and REJ (connection rejected)
12	Missed_bytes	-	Number of missing bytes in content gaps
13	Src_pkts	Spkts	Number of original packets which is estimated from source systems
14	Src_ip_bytes	-	Number of original IP bytes which is the total length of IP header field of source systems
15	Dst_pkts	Dpkts	Number of destination packets which is estimated from destination systems
16	Dst_ip_bytes	-	Number of destination IP bytes which is the total length of IP header field of destination systems

4 EXPERIMENTAL RESULTS AND ANALYSIS

The first step to be able to apply deep learning models is to read and pre-process the raw dataset files. As our data is massive in size, we adopted a special technique to be able to process it according to available resources. As mentioned in section IV our approach is composed of two phases A) Training phase and B) Real time detection phase.

4.1 Training phase

First step is to read the 23 files one by one, apply a pre-processing algorithm, presented next, that is composed of six steps:

1- Changes “src_ip” and “dst_ip” features from string formats into integer representation.

2- Changes “proto”, “service”, “http_trans_depth”, “http_method” and all categorical features that are in string format into integers representing categories. For example HTTP is 0, MQTT is 1 etc.

3- Drops unnecessary or irrelevant features like “http_user_agent”, “ssl_issuer”, “dns_query” etc.

4- Split data into Training features (X) and Target label (Y)

5- Scales the Training features to by removing the mean and scaling to unit variance. At this stage the majority of data in X should be between 4.0 and -4.0. for example, a Source IP feature could have the value of -0.125865. this step is very practical to reduce the resources needed for training and the training time.

6- Appends data to output files and charges next .CSV

ALGORITHM 2: Preprocessing

```
While Dataset folder exist
    List ← dataset file names
    ListIrrelevant ← [Irrelevant Features]
    Data ← Empty dataframe
    Y ← Empty dataframe
    for F in List do
        Temp ← Read(F)
        for Feature is Temp.Columns do
            If Feature is IP then                                   ▷ Step 1
                Temp.Feature ← IntRespresentation(Temp.Feature)     ▷ *
            else if Feature is Category then                        ▷ Step 2
                Temp.Feature ← IntCategorical(Temp.Feature)
            else if Feature in ListIrrelevant then                  ▷ Step 3
                Drop(Temp.Feature)
            end if
        end for
        Y.Append(temp["label"])                                     ▷ Step 4
        Drop(Temp["label"])
        Temp ← Scale(Temp)                                          ▷ Step 5
        Data.Append(Temp)                                           ▷ Step 6
    end for
end while
```

*this is done via the netaddr python library that transforms IP to an unsigned integer representation

At this stage the algorithm will produce two files, X and Y, which are ready to be injected into the deep learning model but as we mentioned earlier the dataset is heavily imbalanced and it is guaranteed to cause an overfitting problem. To overcome this, we have opted to use one of the data balancing techniques. Under-sampling and over-sampling are two conventional approaches to this issue. When the majority class is under sampled, the minority class receives the same quantity of data. But this is inefficient, the data that was destroyed contains crucial details. For this reason, we chose to apply an over-sampling algorithm. Adaptive Synthetic Sampling Approach ADASYN [28] can be used to avoid any of the unbalanced data issues.

The main benefit of the data-generating algorithm ADASYN is producing additional data for "harder to learn" situations and not duplicating minority data. This algorithm is based on K Nearest Neighbours classifier to detect a pre-set number of samples in a virtual space and add copies of these records in the area.

By interpolating between existing instances to create new instances, ADASYN produces synthetic data for the minority class. The algorithm generates more synthetic data in an adaptive manner. In other words, it raises the sample density of examples of the minority class that are located farthest from instances of the majority class.

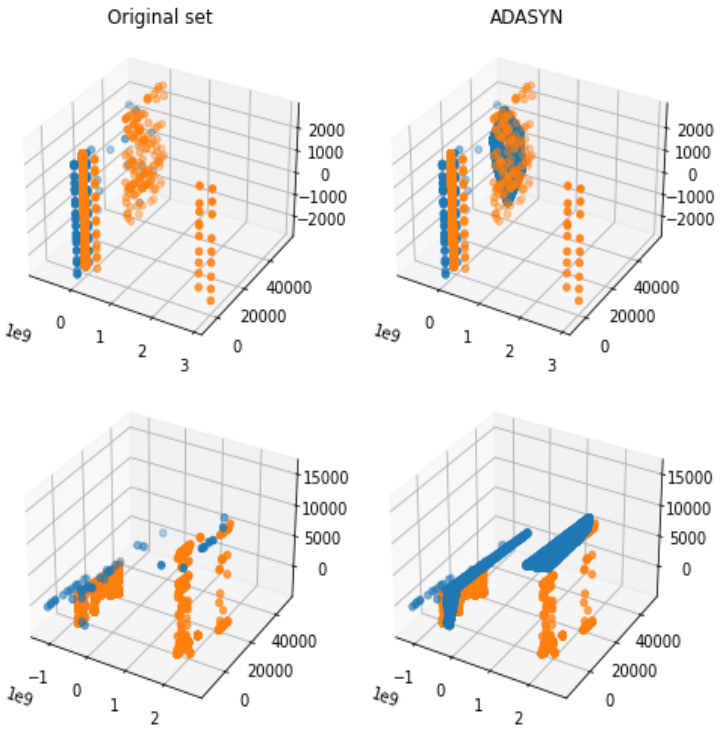

Figure 4: ADASYN results.

Because the resulting data has a big size in the order of multiple gigabytes, we are required to iterate over the X file by batch in order to apply the ADASYN algorithm. We consider each batch as a separate file. We calculate the ratio between classes and we oversample the minority class following the previously explained method as displayed in Fig 4. These figures are produced by applying a PCA algorithm on data and the axes represent the three most significant principal components to be able to visualize data samples in a 3D plan. We see the representation of the original dataset on the left and the oversampled one on the right with the normal packets as orange dots and the malicious packets as blue dots.

In our work we have implemented and evaluated three different models:

1- LSTM-CNN based model applied on original dataset without oversampling in discrete domain.

2- LSTM-CNN based model applied on treated dataset with ADASYN oversampling in discrete domain.

3- LSTM-CNN based model combined with the discussed Profiling component applied on treated dataset with ADASYN oversampling in real time domain.

In the next section we demonstrate the methods of detection we tested for with an extensive results discussion.

4.2 Detection phase

To be able to examine and evaluate the results of the last model, we developed a system that converts the static discrete dataset to a real time packet flow, to imitate real network behaviour. The idea is to test the proposed models with a packet traffic input.

The novelty in our work is to put our model in real time test conditions. Instead of pre-saved data batches being analysed by the proposed deep learning models neglecting any temporal properties of the packets like reception time stamp, duration of exchange and number of packets in a time window, which would dramatically decrease effectiveness of the IDS, we have put in place a system to inject packets according to temporal information from the dataset. The packets are then imitating a real time network traffic.

Figure 5: Dataset to real time conversion.

We variate the time period (t) and the profile difference threshold Delta (Δ) of the profiling component to evaluate the impact of these variables on the performance. The results are demonstrated in table 3.

The packets passing through the network during the pre-set (t) is saved and analysed as a batch taking in consideration the temporal properties and the correlation between packets. Using this approach, the whole batch is classified as attack if only one malicious packet is detected, reducing the FNR to almost zero.

4.3 Tests

In the next section we choose multiple metrics to make an extensive comparison between the models that we propose. Accuracy (AC), false alarm or False Positive Rate (FPR), True Positive Rate (TPR) also called recall (R), precision (P), and F1- score (F1) are the performance evaluation metrics utilized to examine the outcomes of our system. According to the following equation accuracy is calculated as a percentage of the total number of correctly classified packets:

$$AC = \frac{TP + TN}{\mathrm{TP} + \mathrm{FP} + \mathrm{TN} + \mathrm{FN}} \qquad (4)$$

Whereas normal traffic symbolizes negative situations, malicious traffic symbolizes negative instances. The amount of normal traffic that have been successfully identified as legitimate is known as True Positive (TN). The number of malicious instances that have been mistakenly labelled as normal is known as False Negatives (FN). The number of hostile traffic samples that have been successfully identified as anomalous is known as True Positive (TP).

$$FPR = \frac{FP}{\mathrm{FP} + \mathrm{TN}} \qquad (5)$$

True Positive Rate represents the number of malicious samples that were correctly classified.

$$TPR = \frac{TP}{\mathrm{TP} + \mathrm{FN}} \qquad (6)$$

Precision (P) provides the percentage of positively classified samples that are truly positive.

$$P = \frac{TP}{\mathrm{TP} + \mathrm{FP}} \qquad (7)$$

F1-score is a weighted average between precision and recall.

$$F1 = \frac{2 \times P \times R}{\mathrm{P} + \mathrm{R}} \qquad (8)$$

For each version of the model, we log the mentioned metrics for two stages: 1) Learning stage on training data referred as Validation and 2) Classification stage on test data that is not known by the trained model referred to as Test. This is a crucial step to effectively test the stability of our model against real situations.

The percentage of data that is set as Test data is 20% of the total dataset. The remaining 80% is divided into 80% for training and 20% for validation.

Additional performance tests were conducted on the models during detection phase to further investigate the usability of our model in real environments.

We conducted real-time testing by converting static datasets (ToN-IoT and BoT-IoT) into simulated real-time packet flows, accounting for temporal properties like packet timestamps. This approach mimics network behavior by injecting packets according to their original timing data.

We conclude that the LSTM-CNN model without oversampling on data is generating an overfitting because of the big difference in sample numbers between classes

This model is effective only when producing a classification on ten classes representing the attack types, yet it is ineffective with a binary classification between Normal/Attack types.

Table 3: Results

Static Detection tests for TON-IOT						
Model	***Test Data***	***AC%***	***FPR%***	***TPR%***	***P%***	***F1%***
LSTM-CNN without oversampling	*Validation (20%)*	*99.92*	*2*	*98*	*99.92*	*99*
	Test (20%)	*68.52*	*59*	*41*	*65*	*68*
LSTM-CNN with oversampling	*Validation (20%)*	*99.75*	*2*	*98*	*99.7*	*99.7*
	Test (20%)	*98.24*	*3*	*97*	*98.5*	*98.7*
Static Detection tests for BOT-IOT						
Model	***Test Data***	***AC%***	***FPR%***	***TPR%***	***P%***	***F1%***
LSTM-CNN without oversampling	*Validation (20%)*	*89.12*	*9*	*87*	*88*	*87.6*
	Test (20%)	*68.52*	*59*	*41*	*65*	*68*
LSTM-CNN with oversampling	*Validation (20%)*	*97.12*	*2.4*	*97*	*98.2*	*96*
	Test (20%)	*96.84*	*2.8*	*95.3*	*96.5*	*97.7*

Performance tests for TON-IOT			
Model		***Memory (RAM) usage***	***GPU usage***
Static	***LSTM-CNN without Oversampling***	*76%*	*22%*
	LSTM-CNN with Oversampling	*88%*	*31%*
(Real Time) detection	***LSTM-CNN + profiling***	*~0.05%*	-

Performance tests for BOT-IOT			
Static	***LSTM-CNN without Oversampling***	*62%*	*14%*
	LSTM-CNN with Oversampling	*78%*	*26%*
(Real Time) detection	***LSTM-CNN + profiling***	*~0.03%*	-

Table 3 shows the metric for different levels of testing. The first part shows test results on both LSTM-CNN with and without oversampling for the ToN-IoT as well as BOT-IoT datasets. It gives an indication of whether the models are properly trained or whether a over-sampling / under-sampling is occurring. In our case, for the first model, an oversampling problem is clearly visible by comparing the training validation accuracy and the testing accuracy.

We overcome this issue by using the oversampling technique, for the second model. We obtain a much closer validation and testing accuracy for both datasets, as represented in Table 3

The second part of results show that our profiling component reduces GPU usage to near zero in real-time mode by filtering 80–90% of traffic, allowing the Deep Learning model to focus only on suspicious packets. Additional metrics, such as an average detection latency of 13.7 ms per batch and throughput of ~7,300 packets/second, demonstrate practical applicability.

As showed in Fig 6, we are also comparing the impact of varying both time interval (t) and the index delta (Δ) of the proposed profiling model. From the obtained metrics we understand that Delta has a very important effect on the

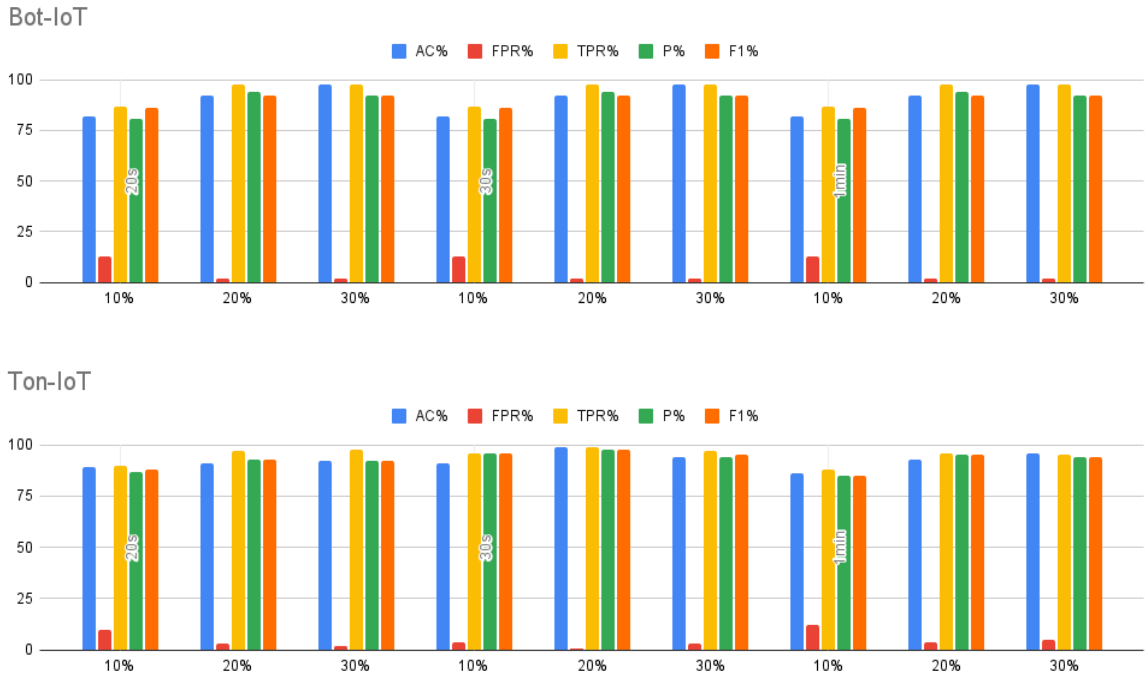

Figure 6: model results

performance regardless of the time interval. Furthermore, we inspected the accordance between time interval and the detected class (attack type), we discover that for each class a certain time interval is optimal which is different from other classes. At this point we are obliged to choose a time interval with the best median value of accuracy. We followed this method because median, unlike average, is biased to the center of highest values meaning that extreme low or high values will have less weights thus not affecting the result.

We also conducted a comparative analysis (Table 4 below) against standalone models (CNN, LSTM and SVM) and state-of-the-art hybrid approaches (e.g., [4], [10]). Our two-stage model achieves comparable accuracy (98.24% on ToN-

IoT) while reducing computational overhead by 40–60% compared to [4] and [10]. The dynamic profiling engine significantly lowers input data volume, enabling efficient deployment on IoT devices.

Table 4: Comparative Performance

Model	Accuracy (%)	FPR (%)	GPU Usage (%)	RAM (MB)
Proposed (LSTM-CNN)	98.24	3.0	0	50
Reference [4]	99.52	0.14	45	780
Reference [10]	97.61	2.5	32	650
Standalone CNN	94.12	5.8	25	320
Standalone LSTM	92.34	6.2	22	290
Standalone SVM	89.5	8.9	36	280

4.4 Discussion

We have presented a model for intrusion detection based on three main aspects: Two stage detection, profiling, and machine learning. As already presented, our architecture begins with a profiler component that is lightweight and ready to be implemented in real world scenarios. This component is based on statistical computing thus it does not need a lot of resources to operate. We intentionally implemented such algorithm in order to allow the real time application of our approach. In the tests we give a brief yet important idea about the RAM and GPU usage of each configuration of our model. We conclude at first that oversampling technique is a must, because the huge gap between training validation and model testing results.

Using ADASYN the LSTM-CNN ML algorithm has reached satisfying results, yet looking at the performance results in the next part of table 3 we discover that it is far away from being suitable for IoT implementation by itself.
The same results also confirm that when combined with the proposed profiler component a huge drop in resource consumption is noticed making this a valuable contribution in our paper.

By varying the parameters of the profiler, we can tune the performance of the detection algorithm to its finest depending on the actual application and context of the IoT system. Yet because of the nature of the statistical formula we propose, the profiler itself is considered as dynamic which means that it will adjust and adapt to normal network flow of packets and automatically change the reference profile. This gives an advantage in case of periodical usage of IoT network which is also a main contribution of our paper.

While transformers excel in sequential data, their computational demands (e.g., 300 MB RAM for BERT) are prohibitive for IoT. Future work could explore lightweight transformers (e.g., MobileViT) paired with our profiling engine to balance accuracy and efficiency.

5 CONCLUSION

In this paper we propose a novel method for detecting intrusions in IoT networks. Through the paper we explain the multiple types of attacks that could occur on IoT networks, putting the light on types, effects and IoT targeted layers. We conduct a detailed study on state of art techniques, algorithms and published works discussing Intrusion Detection Systems. A very important issue we discover in most of the state of art papers is the testing method. If we are presenting an IDS a

real time testing method is crucial. Most of discussed papers work only on static datasets. Further ahead we compare a variety of datasets used in these papers, we state that some of it does not directly relate to IoT but general networking datasets. We carefully choose two datasets ToN-IoT and BoT-IoT as they are the most convenient to our application. We deeply analyze and discuss the different aspects of both datasets (statistical, data nature, attacks presented etc.). From the previous analyze we deduce that both datasets are heavily imbalanced, and our model will certainly result in overfitting which is confirmed later. To overcome this, we use ADASYN technique to oversample the minority class and produce a new balanced dataset.

At this point we propose a model composed of two parts: 1) profiling component 2) classification component. The 1st component is much improved version of an existing paper based on calculating packets traffic statistics and important data fields. The second is a proposed combination between one dimensional Convolutional Neural Networks and Long Short-Term Memories networks.

The results are satisfying comparing to the state of art where we have achieved best performance with less resources and using real-time testing method.

All the codes and documentations are available on www.github.com/kalboussik/

ACKNOWLEDGMENTS

This work is financed by LAMIH UMR CNRS 8201 (Laboratoire d'Automatique, de Mécanique et d'Informatique Industrielles et Humaines) in collaboration with MIRACL : Multimedia, InfoRmation systems and Advanced Computing Laboratory.

REFERENCES

[1] R. Shire, S. Shiaeles, G. Bendiab, B. Ghita, and N. Kolokotronis. 2019. Malware Squid: A Novel IoT Malware Traffic Analysis Framework Using Convolutional Neural Network and Binary Visualisation. In *International Conference on Next Generation Wired/Wireless Networking*. Springer. https://doi.org/10.1007/978-3-030-30859-9_6.

[2] G. Bendiab, S. Shiaeles, A. Alruban, and N. Kolokotronis. 2020. IoT malware network traffic classification using visual representation and deep learning. In *2020 6th IEEE Conference on Network Softwarization (NetSoft)*. IEEE, 444–449.

[3] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour. 2019. Intrusion detection systems in the internet of things: A comprehensive investigation. *Comput. Netw.* 160, 165–191.

[4] N. Ansar, M. S. Ansari, M. Sharique, A. Khatoon, M. A. Malik, and M. M. Siddiqui. 2024. A cutting-edge deep learning method for enhancing IoT security. *arXiv*. https://arxiv.org/abs/2406.12400.

[5] X. Dong, X. Zhang, Y. Sun, L. Chen, M. Yuan, and S. Wang. 2025. SmoothGNN: Smoothing-aware GNN for Unsupervised Node Anomaly Detection. Proceedings of the ACM on Web Conference 2025. Association for Computing Machinery, New York, NY, USA, 1225–1236. https://doi.org/10.1145/3696410.3714615

[6] S. Altamimi and Q. Abu Al-Haija. 2024. Maximizing intrusion detection efficiency for IoT networks using extreme learning machine. *Discover Internet Things* 4, 5. https://doi.org/10.1007/s43926-024-00060-x.

[7] A. Shaikh and P. Gupta. 2022. Real-time intrusion detection based on residual learning through ResNet algorithm. *Int. J. Syst. Assur. Eng. Manag.* https://doi.org/10.1007/s13198-021-01558-1.

[8] H. Altunay and Z. Albayrak. 2023. A hybrid CNN + LSTM-based intrusion detection system for industrial IoT networks. *Eng. Sci. Technol. Int. J.* https://doi.org/10.1016/j.jestch.2022.101322.

[9] Danish Javeed, Tianhan Gao, Muhammad Shahid Saeed, Prabhat Kumar, Randhir Kumar, and Alireza Jolfaei. 2023. A Softwarized Intrusion Detection System for IoT-Enabled Smart Healthcare System. ACM Transactions on Internet Technology. DOI: 10.1145/3634748.

[10] F. Louati, F. Ktata, and I. Amous. 2023. An efficient real-time intrusion detection system for big data environment. In *15th International Conference on Agents and Artificial Intelligence*. https://doi.org/10.5220/0011885900003393.

[11] B. Giampaolo, A. Giuseppe, C. Domenico, P. Valerio, and P. Antonio. 2020. A hierarchical hybrid intrusion detection approach in IoT scenarios. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. https://doi.org/10.1109/globecom42002.2020.9348167.

[12] I. Mutambik. 2024. An efficient flow-based anomaly detection system for enhanced security in IoT networks. *Sensors* 24, 22, 7408. https://doi.org/10.3390/s24227408.

[13] B. Giampaolo, A. Giuseppe, C. Domenico, M. Antonio, P. Valerio, and P. Antonio. 2023. Network anomaly detection methods in IoT environments

via deep learning: A fair comparison of performance and robustness. *Comput. Secur.* 128, 103167. https://doi.org/10.1016/j.cose.2023.103167.

[14] S. Smys, A. Basar, and H. Wang. 2020. Hybrid intrusion detection system for Internet of Things (IoT). *J. IoT Soc. Mob. Anal. Cloud* 2, 190–199.

[15] M. Rashid, J. Kamruzzaman, M. Hassan, T. Imam, and S. Gordon. 2020. Cyberattacks detection in IoT-based smart city applications using machine learning techniques. *Int. J. Environ. Res. Public Health* 17, 9347.

[16] R. Du, S. Li, and P. Zhao. 2024. A behavioral recognition-based federated learning framework for IoT environments. In *2024 International Joint Conference on Neural Networks (IJCNN)*, Yokohama, Japan, 1–9. https://doi.org/10.1109/IJCNN60899.2024.10651433.

[17] P. Krishnan, K. Jain, R. Buyya, P. Vijayakumar, A. Nayyar, M. Bilal, and H. Song. 2022. MUD-based behavioral profiling security framework for software-defined IoT networks. *IEEE Internet Things J.* 9, 9, 6611–6622. https://doi.org/10.1109/JIOT.2021.3113577.

[18] M. Dilraj, K. Nimmy, and S. Sankaran. 2019. Towards behavioral profiling-based anomaly detection for smart homes. In *TENCON 2019 - 2019 IEEE Region 10 Conference (TENCON)*, Kochi, India, 1258–1263. https://doi.org/10.1109/TENCON.2019.8929235.

[19] J. Rose, M. Swann, G. Bendiab, S. Shiaeles, and N. Kolokotronis. 2021. Intrusion detection using network traffic profiling and machine learning for IoT. In *IEEE 7th International Conference on Network Softwarization (NetSoft)*. https://doi.org/10.1109/NetSoft51509.2021.9492685.

[20] A. Jahromi, H. Karimipour, A. Dehghantanha, and K. Choo. 2021. Toward detection and attribution of cyber-attacks in IoT-enabled cyber-physical systems. *IEEE Internet Things* 8, 13712–13722.

[21] KDD Cup 1999. Retrieved March 18, 2024, from https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.

[22] 1998 DARPA Intrusion Detection Evaluation Dataset. Retrieved March 18, 2022, from https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset.

[23] G. Mohi-ud-din. 2018. NSL-KDD. *IEEE Dataport*. https://doi.org/10.21227/425a-3e55.

[24] N. Moustafa and J. Slay. 2015. 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)*, Canberra, ACT, Australia, 1–6.

[25] I. Sharafaldin, A. Lashkari, and A. Ghorbani. 2018. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *4th International Conference on Information Systems Security and Privacy (ICISSP)*, Madeira, Portugal, 108–116.

[26] A. Alsaedi, N. Moustafa, Z. Tari, A. Mahmood, and A. Anwar. 2020. TON IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access* 8, 165130–165150.

[27] N. Koroniotis, N. Moustafa, E. Sitnikova, and B. Turnbull. 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. *Future Gener. Comput. Syst.* 100, 779–796.

[28] H. He, Y. Bai, E. Garcia, and S. Li. 2008. ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *International Joint Conference on Neural Networks*.TUG 2017. Institutional members of the LaTeX Users Group. Retrieved May 27, 2017 from http://wwtug.org/instmem.html