# IoT ANOMALY DETECTION WITH 1D CNN USING P4 CAPABILITIES

Gereltsetseg ALTANGEREL*, Máté TEJFEL**, Enkhtur TSOGBAATAR***
* ** Department of Programming languages and Compilers, Faculty of Informatics, Eötvös Loránd University (ELTE), Budapest, Pázmány Péter stny. 1/C, 1117, Tel. 061 372 2500, E-mail: gereltsetseg@inf.elte.hu*, matej@inf.elte.hu**
ORCID: 0000-0002-1594-8158*, 000-0001-8982-1398**
*** Department of Information Network and Cyber Security, Mongolian University of Science and Technology (MUST), Ulaanbaatar, Baga toiruu 34, Sukhbaatar 14191, Tel. 976 11 323579 E-mail: enkhtur@sict.edu.mn

**ABSTRACT**

*Although the Internet of Things (IoT) is a rapidly developing technology, it also brings a number of security challenges, such as IoT attacks. Currently, research on IoT anomaly detection in Software-Defined Networking (SDN) relies only on the control plane. In this study, we aim to detect IoT anomalies by covering the advantages of the control and data plane. First, we collected real-time network telemetry data from the data plane based on the capabilities of the P4. Then, using this telemetry data, we built different anomaly detection models and compared their performance. Among them, the one-Dimensional Convolutional Neural Network (1D CNN) model classified our data best and showed the highest performance, so we proposed this model for IoT anomaly detection on the control plane. To our knowledge, our approach is the first solution that integrates the control plane and data plane for IoT anomaly detection. Finally, when evaluating the performance of our proposed 1D CNN model, the accuracy, F1 score, and Matthews correlation coefficient (MCC) are the same or better than existing studies.*

***Keywords:*** *P4, In-Band Network Telemetry(INT), IoT security, 1D-CNN, SDN*

## 1. INTRODUCTION

With the exponential increase in the number of IoT devices worldwide and the ease with which these vulnerable IoT devices can be exploited, the flow of IoT-based attacks has reached unprecedented levels [3, 19]. For example, the Mirai Botnet launched a distributed Denial of Service (DDoS) attack against the Dyn DNS provider and took down several sites, including GitHub, Reddit, Netflix, and Airbnb. Due to the complexity of securing IoT devices, network-based anomaly detection methods based on artificial intelligence/machine learning (AI/ML) techniques are widely used for IoT anomaly detection.

The emergence of an innovative SDN networking paradigm [25] has made AI/ML-based anomaly detection solutions more flexible and efficient. Many anomaly detection solutions based on AI/ML techniques have been proposed by leveraging the traffic flow control and monitoring capabilities of SDN [14]. Therefore, IoT anomaly detection solutions are often based on the control plane.

In this research, we aim to detect IoT anomaly detection by covering the capabilities of control and data planes.

First of all, we collected real-time network telemetry information directly from the data plane by leveraging the P4 capability. P4 is a language for programming algorithms in the data plane, and many useful applications are being developed with it, one of which is In-Band Network Telemetry (INT), used for collecting telemetry information. As far as we know, currently, network datasets (netflow, wireshark) are mainly used for anomaly detection. Compared to the network dataset, the telemetry data collected from the data plane is more advantageous because it is real-time and will make detection more efficient. We collected telemetry data during the normal, DDoS and very short interval-DDoS (VSI-DDoS) attack conditions and made public datasets for future research. Shan et al. [29] present a new burst of low-rate DDoS attacks known as VSI-DDoS attacks. Because this attack mimics legitimate user behavior, it is difficult to detect compared to other attacks, so we chose this attack to test our proposed model.

After that, based on the collected telemetry data (testbed dataset), we designed a 1D CNN-based IoT anomaly detection model that can be implemented on an SDN controller. AI/ML-based anomaly detection involves first training the system on a normal baseline and then comparing the activity to that baseline [26]. This kind of anomaly detection is widely used to detect IoT attacks and network attacks. We also used this paradigm in our study.

Finally, we perform a systematic and extensive experimental analysis using testbed and benchmark datasets and show that the proposed 1D-CNN outperforms its competitors and other commonly used algorithms in terms of accuracy, F1, and MCC metrics.

To our knowledge, our approach is the first solution that integrates control plane and data plane capabilities for IoT anomaly detection. The rest of the paper is structured as follows: Section 2 discusses related works, Section 3 provides background information, Section 4 explains the setup of our experimental network and telemetry data collection, Section 5 presents our proposed model, Section 6 details the experimental results and discussions, and finally, in Section 7, we summarize our findings and suggest future directions.

## 2. RELATED WORK

Numerous research works have been done on IoT anomaly detection at the network level. These studies can generally be categorized by whether they were conducted in SDN and traditional network environments or by what AI/ML methods were used.

The first category of IoT anomaly detection methods was based on machine learning algorithms [4,8]. For example, Bhunia and Gurusamy [4] present a machine learning-based anomaly detection and mitigation method on IoT traffic using SDN. It uses the SVM algorithm to monitor

and learn the behavior of IoT devices over time, and to detect anomalies.

The next category can be deep-learning-based anomaly detection approaches. In recent years, advances in deep learning have revolutionized many areas of data-driven modeling [13, 20, 21, 24]. Meiden et al. [21] presented a network-based IoT anomaly detection method using deep autoencoders. The method is validated using nine different commercial IoT devices in a laboratory environment with two popular IoT-based botnets, Mirai and BASHLITE.

The third category [7, 30, 31] has used ensemble learning models to improve anomaly detection performance. Because ensemble learning uses multiple algorithms, it can produce better prediction performance than any single algorithm. For example, Timcenko and Gajin [30] presented an IoT anomaly detection method using an ensemble learning approach and achieved 99% accuracy using the UNSW-NB15 dataset.

Another category of IoT anomaly detection methods exploits the advances and flexibility in SDN networks [4, 12, 31]. Del-IoT [31] is an SDN-based IoT anomaly detection approach using a deep ensemble model. The solution is implemented on an SDN controller and uses SDN flexibility such as dynamic flow control in the presence of attacks and device state prediction based on status tables.

Most of the research work not only detects anomalies at the network level but also considers and solves different types of problems such as data imbalance, data heterogeneity, dynamic flow management, and mitigation of the attack. In addition, current research shows that controller-based anomaly detection methods are popular in SDN networks, and almost all of these solutions use AI/ML-based techniques to detect anomalies using network traffic datasets (e.g., netflow, wireshark).

Research on anomaly detection in SDN environments exploiting data plane capabilities is relatively scarce compared to controller-based solutions. Also, anomaly detection experiments in the data plane are currently not very common. However, there are some studies and experiments in this field. Musumeci et al. [22] implemented machine learning-based anomaly detection in P4 programmable network. They implemented traffic feature extraction on the P4 switch and forwarded them to the anomaly detection module. Euclid [15] is DDoS data plane detection solution that uses the Shannon entropy based on the frequency of source and destination Internet Protocol (IP) addresses. Sanghi et al. [27] try to identify possible attacks on data-plane systems; and designing a scalable tool for detecting such attacks at run time. Kang et al. [16] propose an approach to discovering attack vectors in a data plane system and present a set of preliminary experiments to demonstrate the feasibility of this approach. These are the first attempts to detect anomalies in the data plane and none of them use INT data.

Almost all network anomaly detection methods use datasets based on network traffic, and the telemetry information collected from the network in real-time is limited. As far as we know, only one study [23] has used INT data for anomaly detection. It applied a recurrent neural network (RNN) model, but its performance was not high enough compared to the above models.

## 3. BACKGROUND

### 3.1. SDN overview

Network packet processing algorithms are divided into control and data planes by function. Data plane algorithms define the packet processing pipeline on the device, while control plane algorithms define the data forwarding rules in the data plane.

The SDN separates the control plane from the data plane, and control plane algorithms run on a dedicated controller platform (e.g., server). The idea of programmable networking re-gained momentum with the emergence of the SDN paradigm. It promises to dramatically simplify network management and enable innovation through network programmability [25]. Fig. 1 shows the interaction between the control and the data plane, where the SDN controller (control plane) communicates with the network infrastructure (data plane) through a southbound Application Programming Interface (API).

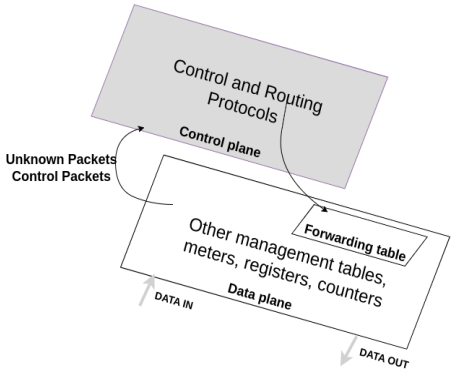



**Fig. 1** Control & Data plane interaction

### 3.2. AI/ML-based anomaly detection with SDN

With the global view of the network and control capabilities of SDN, network security can be improved by using anomaly detection modules as applications on top of the controller. Integrating security features in this way is not complicated [14]. There are solutions for network anomaly detection using SDN capabilities, many of which have been implemented using AI/ML or statistical methods [2, 6, 9, 33]. These solutions make efficient use of the SDN control plane, for example, when problematic traffic is detected, the control plane can trigger appropriate actions. Therefore, the SDN control plane has made AI/ML-based anomaly detection solutions more flexible and efficient.

### 3.3. The programmable data plane and P4

The programmable data plane is a relatively recent development in computer networking when compared to the programmable control plane. Unlike before, where only device manufacturers could program it, the programmable

data plane now enables anyone to rapidly design, test, and deploy diverse applications directly on the data plane. P4 is one of the most popular domain-specific programming languages for defining data plane algorithms.

Fig. 2 shows the basic packet processing architecture of P4, including a parser, match-action, and deparser. The parser receives incoming packets and extracts the header fields from them. Then, the match-action section processes the packet headers and the metadata of the packets. It contains one or more tables, and the table is the most fundamental unit of the match-action section. The tables have a key part and an action part, and during the table application, the program tries to find the most appropriate key in the table based on the header fields and the metadata of the actual packet and executes the action part belonging to that key. If there is no appropriate key, the program executes the default action if it is defined, or does nothing. For example, considering the most commonly used routing table, the match key is the destination IP address, the match type can be the longest prefix match (can be different) and the corresponding action depends on the match result: whether to forward the packet or drop, or no-action. The deparser builds the outgoing packet by assembling the processed header fields and the original payload [5].

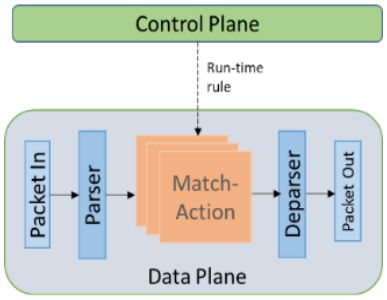



**Fig. 2** Abstract packet forwarding in P4.

The programmability of the data plane allows for the development of many interesting applications [1]. One of these is a P4-based monitoring system called In-Band Network Telemetry (INT), which collects network telemetry information (hop latency, flow latency, queue depth, etc.) from the data plane. Since this process does not involve a CPU-driven control plane, telemetry data (INT data) is collected in more real-time compared to conventional monitoring mechanisms. This telemetry data is collected using a standard packet over a network or a probe packet (INT packet). Each node that receives an INT packet embeds telemetry data in this packet. The source INT switch adds INT instructions (header) to the packet on what to collect from the devices, and then that packet is transmitted through the network for collecting INT data. Finally, the edge device (INT sink switch) removes the INT header and INT data from the packet, and INT data is transferred to the INT collector [17].

## 4. CREATING INT DATASET

### 4.1. INT enabled testbed network

We created the SDN network shown in Fig. 3 on the MININET[1] simulation program. This SDN network consists of a data plane, a control plane, IoT servers, and users. There are INT-enabled P4 software switches (bmv2[2]) with redundant connections in the data plane, and ONOS[3] controller in the control plane is responsible for controlling the P4 switches and instructing what type of INT data to collect. The knowledge plane collects INT data from the network and learns and detects anomalies based on the INT data. Therefore, it includes an INT collector and our proposed 1D-CNN IoT anomaly detection module introduced in Section 5. This detection module can be integrated into the control plane as an application. The INT collector is a high-performance INTCollector introduced by Van Tu et al. [32].

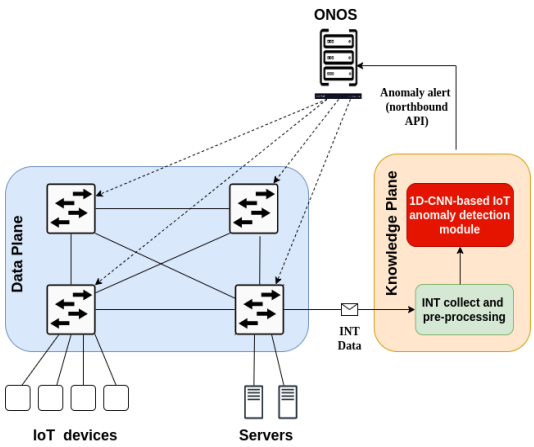



**Fig. 3** SDN testbed network with IoT anomaly detection module

We simulated external IoT devices as attacker nodes and one of the servers as target nodes. The mirror port is configured on the edge switch connected to the INT collector; this edge switch sends all INT data (from IoT users to the target server) to the INT collector. The INT collector filters the valuable information from the data and stores it in the influx[4] database. The next section explains how the INT dataset was generated during the attacks and under normal conditions using this network setup.

### 4.2. INT data collection on testbed network

Firstly, we have collected INT data, including the protocol, port number, flow latency and hop latency of each intermediate node in a normal traffic state. In this state, we

[1] http://mininet.org/
[2] https://github.com/p4lang/behavioral-model
[3] https://opennetworking.org/onos/
[4] https://www.influxdata.com/
[5] https://iperf.fr/
[6] https://linuxize.com/post/linux-ping-command/

generated legitimate traffic, including User datagram protocol (UDP) and Internet Control Message Protocol (ICMP) flows using iperf[5] and ping[6] tools from external IoT devices to the target server and saved INT data in influx database at INT collector. We collected more than 200 records of data in an hour under this normal condition.

Secondly, INT data is collected in malicious traffic conditions. Due to hardware limitations, we chose the ICMP flooding-based DDoS attack and created two versions of the DDoS attack.

**Scenario 1 - DDoS with ICMP flooding:** We created a DDoS attack, and legitimate traffic flows from the attacker node to the target node. The DDoS attack was generated using the hping3 tool[7], and INT data were collected for one hour under this condition.

**Scenario 2 - VSI-DDoS:** VSI-DDoS [29] is a new class of DDoS attacks with short-burst and low-rate designed to degrade the quality of service(QoS). This attack is more challenging to detect than a typical DDoS because it tries to mimic the legitimate behavior of the traffic. Therefore, it is more appropriate to test the performance of the proposed anomaly detection model. INT data were also collected for one hour under this condition.

Over 2 hours of data collection, including normal and attack conditions, resulted in an overheating issue with the testbed hardware. Therefore, we used Synthetic Minority Oversampling Technique (SMOTE) [8] to increase the number of collected samples. It augments the data by generating random samples based on the original data's distribution without compromising the data's underlying characteristics. With this, we increased the 431 records in the original DDoS dataset to 104,547 and the 533 records in the VSI-DDoS dataset to 155,325. The normal-to-malicious traffic ratio is around 70x30 for both datasets. When we checked the original dataset and the oversampled dataset using descriptive statistics, the oversampled dataset kept the distribution of the original dataset.

### 4.3. Experimental datasets and their characteristics

In addition to the testbed dataset collected above, we used benchmark datasets to validate our proposed IoT anomaly detection module. Although there is a publicly available INT dataset [9], it did not meet our requirements as it was unlabeled, had few features and so on. Hence, we used the NF-BoT-IoT and NF-ToN-IoT datasets for the initial empirical validation of our proposed model. These IoT NetFlow-based datasets were created at the University of Queensland and are a widely used and efficient dataset for research [28].

NF-BoT-IoT is built using BoT-IoT, and NF-ToN-IoT is built using ToN-IoT datasets. The features on NF-BoT-IoT were extracted from the publicly available pcap files and the flows were labeled with their respective attack categories, but we only considered DDoS attacks. NF-ToN-IoT used the publicly available pcaps of the ToN-IoT dataset to generate NetFlow records. Therefore in our experiment, we used four kinds of datasets:

**Testbed (DDoS):** This dataset contains 73,185 legitimate and 31,362 malicious samples, the features of this dataset are hop latencies of every switch, flow latency, and protocol.

**Testbed (VSI-DDoS):** This dataset contains 104,833 legitimate and 45,167 malicious samples, the features of this dataset are the same as the previous one.

**Benchmark (NF-BoT-IoT):** This dataset contains 13,860 legitimate and 56,840 malicious samples, the features of this dataset are l4_src_port, l4_dst_port, protocol, l7_pro, in_bytes, out_bytes, in_pkts, out_pkts, tcp_flags, and flow_duration

**Benchmark (NF-ToN-IoT):** This dataset contains 270,279 legitimate and 326,345 malicious samples, the features of this dataset are same as NF-BoT-IoT.

All collected features in this dataset are network-specific properties. To observe the behavior of these testbed and benchmark datasets, we calculated the cumulative distribution function of legitimate and attack cases in these datasets.

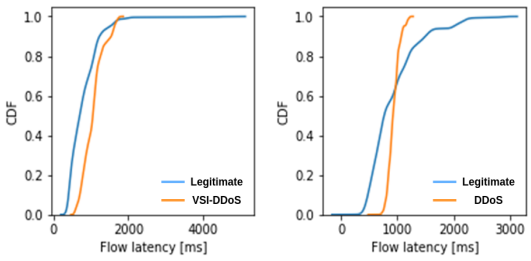



**Fig. 4** Testbed datasets

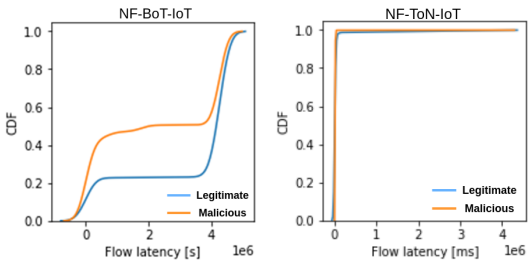



**Fig. 5** Benchmark datasets

The general shape of malicious and legitimate data in VSI-DDoS is relatively similar. In contrast, the malicious and legitimate data in the DDoS dataset are relatively different in Fig. 4. This proves that VSI-DDoS mimics regular traffic.

In Fig. 5, the behavior of legitimate and attack flows in the benchmark datasets is also relatively similar in the flow latency feature. Some studies mention that normal and malicious flows tend to have relatively similar distributions [6].

[7] https://www.kali.org/tools/hping3/
[8] https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/
[9] https://github.com/obiwan96/AnomalyDetection-with-INT

Although the shape of the graphs on the benchmark dataset is somewhat similar, there is a difference in size for NF-BoT-IoT.

These behaviors allow us to make the first hypothesis that the model that can detect anomalies well on the VSI-DDoS and benchmark datasets can perform well on the DDoS dataset. Therefore, we first considered the performance of the more complex VSI-DDoS dataset when choosing our model.

## 5. PROPOSED 1D CNN-BASED ANOMALY DETECTION MODEL USING INT DATA

Based on our dataset's behavior, the problem's nature, recent neural network trends, and several experiments, we found a CNN-based model to be the most suitable for our framework's anomaly detection module. The following sections describe the proposed model and experimental results in more detail.

### 5.1. 1D CNN-based model

The traditional CNN model tended to stack convolution layers and pooling layers like a sandwich, but nowadays, the trend is to stack several convolution layers with small filters [11].

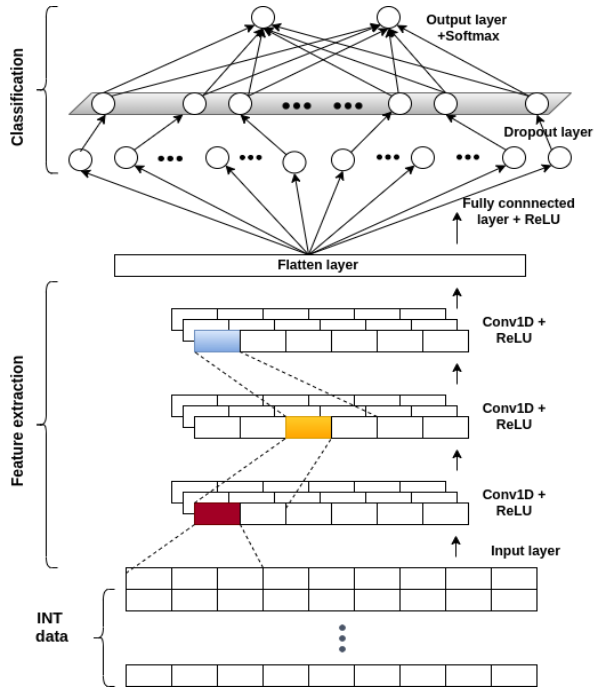



**Fig. 6** 1D CNN-based model

Primarily used in computer vision, these design trends reduce the number of trainable parameters and make the model more cost-effective. Here, we tried to apply these concepts to our target model, and based on our experiments, we built a model that best describes our INT data, which we named the 1D CNN-based model. The architecture of our model is shown in Fig. 6.

First, we built a feature extraction stage that includes three convolutional layers with small filters and Rectified Linear Unit (ReLU) activation function to better identify local patterns from the INT input data. Compared to the fully connected (FC) layer and the long-short-term memory (LSTM) layer, the convolution layers are arranged in this way, allowing feature extraction to be performed more efficiently with fewer parameters [11]. The output of the last convolution layer is then transformed through a flatten layer into a single vector to feed the FC or dense layer in anomaly classification. After that, we stacked two dense layers to form the anomaly classifier, and we put a dropout layer to prevent overfitting[10] problem between these two layers.

**Feature extraction:** One-dimensional convolution layers (Conv1D) is a special form of CNN used to process sequential data. Dense layers learn global patterns in their input feature space, whereas convolution layers learn local patterns from data, as mentioned above. To better understand this, let us look at how image patterns are learned with several convolution layers: A first convolution layer will learn small local patterns such as the edges of the image; a second convolution layer will learn larger patterns made of the features of the first layers, and so on. This process allows convnets to learn data patterns efficiently, and the process at the convnet layer is called convolutional operation, a type of linear transformation.

The convolution layer comprises several convolution kernels, which are used to compute different feature maps. The new feature map can be obtained by convolving the part of the input with a kernel vector. All spatial locations of the input share the kernel according to the stride, resulting in different feature maps. Stride is a hyperparameter of the neural network's kernel that determines the amount of movement over the input data. The complete feature maps are obtained by using several different kernels. Mathematically, the feature value at location (i, j) in the $k^{th}$ feature map of $l^{th}$ layer, $z^l_{i,j,k}$, is calculated by:

$$z^l_{i,j,k} = w^{l\,T}_k * x^l_{i,j} + b^l_k \tag{1}$$

where $w^l_k$ and $b^l_k$ are the weight vector and bias term of the $k^{th}$ filter of the $l^{th}$ layer respectively, and $x^l_{i,j}$ is the input patch centered at location (i, j) of the $l^{th}$ layer.

Then, the ReLU activation function f is applied on a convolutional feature ($z^l_{i,j,k}$), resulting in new output feature ($a^l_{i,j,k}$). The non-linear transformation of the ReLU helps to learn complex patterns in the data.

$$a^l_{i,j,k} = f(z^l_{i,j,k}) \tag{2}$$

where, the ReLU activation function as follows:

$$f(z^l_{i,j,k}) = \begin{cases} z^l_{i,j,k}, \text{ if } z^l_{i,j,k} > 0 \\ 0, \text{ if } z^l_{i,j,k} \leq 0 \end{cases} \tag{3}$$

Therefore, the linear convolution operation and nonlinear ReLU functions on the Conv1D layers form a 1D CNN

[10] https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

computational node that can extract valuable features from our INT input data. Then, these features are sent to the anomaly classification stage, which produces the final results.

**Anomaly classification:** Our model constructs an anomaly classifier using two fully connected layers. While a node in a convolutional layer is only connected to "nearby" nodes from the preceding layer within the width of the convolutional kernel, a node in a fully connected layer is connected to every node in the preceding layer. The Flatten layer converts the output of the final convolutional layer into a single one-dimensional vector, that can be used as input for the first FC layer.

Both fully connected layers also have linear and non-linear transformations (with activation function). The first dense layer with the ReLU activation function is responsible for exploring the global input pattern. The input of this dense layer is the output of the flattened layer, or the input feature vector $\vec{x} = \{x_0, x_2, ....x_{n-1}, x_n\}$, and all these features are fed to each computational node at the dense layer to perform the following linear function. For example, the operation on $j^{th}$ node is:

$$z_j = \vec{x} \cdot \vec{w}_j + b_j \tag{4}$$

where $\vec{w}_j$ is weight vector, $b_j$ is bias at the $j^{th}$ node, $\vec{x}$ is the input feature vector, and $\cdot$ is dot product operation.

For example, in our case, the first dense layer has 16 neurons or computational nodes, the output features of this dense layer are $\vec{z} = \{z_1, z_2, z_3...z_{16}\}$ according to the calculation above, and then each output will be transformed by the ReLU function and it creates $\vec{z'} = \{z_1^{'}, z_2^{'}, z_3^{'}...z_{16}^{'}\}$ output. The non-linear transformation of the ReLU function appears in (3).

The second dense layer with softmax activation and two nodes are dedicated to classification: one node identifies legitimate and the other malicious traffic. The input to this layer is the output of the first dense layer.

The linear transformation at the node of this layer is the same as (4), but the activation function is different (softmax). The softmax function correlates the outputs in two nodes. That means that the softmax function transforms a vector of y real values into a vector of $y^{'}$ real values whose sum is equal to 1. The input values resulting from the linear transformation in the dense layer can be any number. However, softmax transforms them to a value between 0 and 1 to interpret them as probabilities.

$$y_i^{'} = \sigma(y_i) = \left( \frac{e^{y_i}}{\sum_{j=1}^{m} e^{y_j}} \right) \tag{5}$$

where $\sigma$ is the softmax function and m is the number of output values from the dense layer.

Let's see normalization and probabilities with $\sigma(y_i)$ in the example of our model. The output layer has two nodes, so if the output of the first node is $y_1 = -1$ and the output of the second node is $y_2 = 0.5$, softmax converts these numbers into probabilities $\sigma(-1) = 0.18$ and $\sigma(0.5) = 0.82$. The second output determines whether it is legitimate in our case, and since the probability is 0.82, the IoT traffic can be considered legitimate.

The number of layers, their configuration, the activation functions, loss, optimization functions, and hyperparameters of our model were selected based on our experimental results. The experimental results are presented in Section 6.

## 6. EXPERIMENTAL RESULTS AND DISCUSSIONS

### 6.1. Overview of model performance metrics

In this section, we will briefly describe the accuracy, precision, recall, F1 score, and MCC metrics to evaluate the performance of our anomaly detection model. They are computed on the confusion matrix used to illustrate and summarize the performance of the classification algorithm on the test dataset. Since our problem is a binary classification (legitimate vs. abnormal), the confusion matrix is a 2x2 matrix. Therefore, True Negative (TN) and True Positive (TP) represent correctly predicted cases and are located on the matrix's main diagonal. False Negative (FN) and False Positive (FP) are false predictions and lie on the antidiagonal of the confusion matrix. Table 1 shows how these performance metrics are calculated.

**Table 1** Performance metrics

| Metric | Formula |
|---|---|
| **Accuracy** | $A = \frac{TP+TN}{TP+TN+FP+FN}$ |
| **Precision** | $P = \frac{TP}{TP+FP}$ |
| **Recall** | $R = \frac{TP}{TP+FN}$ |
| **F1 score** | $F1 = 2 * \frac{P*R}{P+R}$ |
| **MCC** | $MCC = \frac{TP*TN-FP*FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$ |

The accuracy is the proportion of samples that are correctly classified. Precision measures the accuracy of positive predictions, while recall measures the completeness of positive predictions. F1-score is the harmonic mean of the precision and recall. MCC measures the quality of the classification, showing the correlation agreement between the observed values and the predicted values. We evaluated experimental models by using these metrics.

### 6.2. Overview of machine learning parameters

Learning rate, batch size, the number of epochs, and types of loss functions are primary hyperparameters used to control the machine learning process. In contrast, weights and biases are key parameters learned by the machine itself or adjusted based on the training process. It can also be called trainable parameters.

The loss function checks the validity of the prediction, and the optimization algorithm determines how much and in which direction the trainable parameters would change

based on the output of the loss function. The loss is almost synonymous with error but usually means average error across all items in the training data. Machine learning aims to keep the loss as low as possible (close to zero).

Hyperparameter optimization or tuning is the selection of a set of hyperparameters suitable for a model to maximize performance.

### 6.3. Experimental models, experiments, and evaluation

We built CNN and LSTM models for the INT data with different layers and configurations, and Table 2 summarizes the significant models tested.

**Table 2** CNN and LSTM models

| Model (abbr) | Layer | Kernel conf-s | Neurons | Paras |
|---|---|---|---|---|
| **1D CNN (M1)** | 3 | (3x1)x8, (2x1)x8, (2x1)x8 | - | 482 |
| **1D CNN+Pool (M2)** | 3+2 | (3x1)x8 (2x1)x8, (2x1)x8 | - | 514 |
| **LSTM+Dense (M3)** | 1+1 | - | [32-16] | 20,545 |

First of all, we created various CNN-based models by stacking two or three convolution layers with varying kernel configurations. Among these models, the 1D CNN model (M1) in Table 2 outperformed the others. This model has three convolutional layers with different kernel configurations. In the first row of Table 2 the kernel configuration (3x1)x8 means that the kernel size (weight vector) is 3x1 and the number of kernels in the first layer is 8. The second and the third similar configurations in the row describe the parameters of the second and the third layers correspondingly.

The next model is the 1D CNN+Pool model (M2), where we added a max pooling layer and batch normalization to the M1 model. As mentioned in Section 5, this model has a more traditional layout, while the 1D-CNN model (M1) follows modern trends.

The last model is an LSTM model with one LSTM layer (32 neurons) and one dense layer (16 neurons). The Paras in the last column of Table 2 show the number of trainable parameters. Note that each model uses the same classification stage as in Fig. 6, where we only mention layers in the feature extraction stage.

When testing and building the models with different configurations shown in Table 2, we performed hyperparameter tuning experiments for all models using a grid search algorithm (GSA). Based on this experiment, we obtained the optimal hyperparameters with the highest performance for each model and dataset. That means we varied batch sizes with different learning rates, loss, and optimization functions as follows.

The learning rate defines the learning speed of the model. Therefore, increasing the learning rate increases the learning speed of the model, but at the risk of exceeding the minimum loss. Conversely, if the learning rate is too low, reaching the minimum loss will take longer. When we varied the training rate from 0.0001 to 0.1 with different batch sizes, all models showed the highest performance at 0.001.

Batch size defines how many samples the model uses to estimate the loss at each training iteration (epoch). We set batch sizes of 32, 64, 128, and 256 (power of 2) according to the best practice. The number of epochs varied between 15-30 based on loss values.

For each experiment with different batch sizes and learning rates, we also tested different loss and optimization functions, among which the binary _crossentropy loss function and the Adam optimization algorithm [18] were selected as having the highest performance.

The machine learning experiments have two main phases: training and testing. Working with a dataset collected from a different network would require retraining the model, a common practice in machine learning. Therefore, we conducted separate training and testing procedures for each of the four distinct types of datasets we used. We split each dataset 70x30 according to best practice, 70% of which is used for training the machine and 30% for testing. The models are evaluated based on the testing phase results because the model has not seen the testing dataset before, and the model's performance on such new data is a crucial indicator of the model. Therefore, all performance measurements shown in Table 3 are the testing phase results with the highest performance.

### 6.4. Model selection

We selected the model based on the above experimental evaluations, cost-effectiveness, and other factors.

First of all, we consider performance evaluation. Table 3 shows the performance of our proposed 1D CNN compared with other models. From this table, our model outperforms other datasets in terms of classification quality (MCC), positive prediction accuracy (Pre), positive prediction completeness (Rec), F1, and overall accuracy (Acc). Of these, the classification quality (MCC) is much higher. In addition, the performance on the testbed (VSI-DDoS) and the benchmark (NF-ToN-IoT) is inferior to the performance on the other two datasets because the behavior of normal and malicious traffic is very similar in Fig. 4 and 5. This confirmed our initial hypothesis that the models that performed well on the VSI-DDoS dataset performed well on the other datasets.

Second of all, we consider cost-effectiveness: looking at the number of trainable parameters in Table 2, the 1D CNN (M1) is preferred due to its much smaller trainable parameters. For example, the LSTM-based model learns 20,545 parameters to achieve an accuracy of 99.27 on the VSI-DDoS dataset. Since machine learning models use mathematical operations to find the best parameters to describe the data, our proposed 1D CNN-based model can run faster than LSTM-based models because they consume less memory with fewer parameters. The number of trainable parameters of the M1 model is 42 times smaller than that of LSTM.

**Table 3** Performance evaluation of models on datasets

| Models | Acc | F1 | MCC | Pre | Rec |
|---|---|---|---|---|---|
| **Tesbed (DDoS)** | | | | | |
| 1D CNN | 99.86 | 99.9 | 99.68 | 99.85 | 99.87 |
| CNN+Pool | 99.2 | 99.44 | 98.08 | 99.64 | 99.23 |
| LSTM | 99.7 | 99.79 | 99.3 | 99.68 | 99.68 |
| **Tesbed (VSI-DDoS)** | | | | | |
| 1D CNN | 99.63 | 99.74 | 99.1 | 99.72 | 99.75 |
| CNN+Pool | 99.18 | 99.42 | 98.01 | 99.33 | 99.51 |
| LSTM | 99.27 | 99.48 | 98.22 | 99.24 | 99.73 |
| **Benchmark(NF-BoT-IoT)** | | | | | |
| 1D CNN | 99.92 | 99.95 | 99.75 | 99.96 | 99.94 |
| CNN+Pool | 99.46 | 99.62 | 98.68 | 99.46 | 99.78 |
| LSTM | 99.91 | 99.95 | 99.74 | 99.95 | 99.94 |
| **Benchmark(NF-ToN-IoT)** | | | | | |
| 1D CNN | 99.61 | 99.64 | 99.21 | 99.58 | 99.70 |
| CNN+Pool | 99.20 | 99.45 | 98.08 | 99.58 | 99.32 |
| LSTM | 99.41 | 99.46 | 98.82 | 99.55 | 99.37 |

Then, we considered other factors besides performance and cost-effectiveness. CNN-based models usually use a convolution layer followed by a pooling layer, but we did not want a pooling layer due to the characteristics of our INT data. In the case of multi-dimensional data having many features, the pooling layer has the advantage of extracting more specific features. It also reduces the computational power required to process the data through dimensionality reduction [10]. In contrast, on our data (one-dimensional and time series data with a small number of features), the pooling layer has the disadvantage of removing unique features. Thus, compared to M2 with pooling layer, M1 (built with only convolution layers in feature extraction) had a higher performance for all datasets.

Based on the above considerations, we have chosen 1D CNN-based model (M1) for our INT dataset.

### 6.5. Performance evaluation with competing models

IoT anomaly detection is treated as a machine learning binary classification problem. Therefore, we compared the anomaly detection accuracy of the 1D CNN-based model with well-known classification algorithms such as SVM, Logistic Regression (LR), and Naive Bayes (NB), and Fig. 7 shows the accuracy comparisons. As shown in Fig. 7, our model has higher accuracy than other models for the target VSI-DDoS dataset and other datasets. SVM is the closest to the performance of our model, but the training time was much longer.

Then we compare the overall performance of our model on the VSI-DDoS dataset with the model of Kim et. al. [17] in Table 4. The F1 score of our model is much better than that of the RNN-based neural network in Table 4. Detection accuracy is one of the important parameters for classification problems, but this parameter wasn't considered in the previous study.

**Table 4** Performance comparison with competing model

| Models | Recall | Precision | F1 score |
|---|---|---|---|
| 1D CNN | 99.75 | 99.72 | 99.74 |
| RNN | 100 | 85.9 | 99.41 |

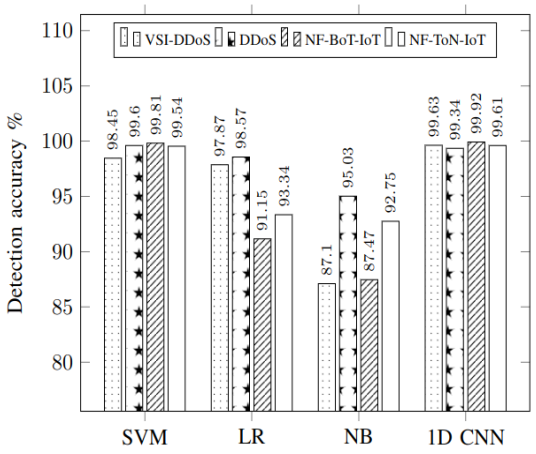



**Fig. 7** Detection accuracy comparison with baseline models

## 7. CONCLUSIONS AND FUTURE WORKS

In this work, we propose a 1D-CNN model for IoT anomaly detection using data plane and control plane capabilities on SDN. The telemetry data we collect from the P4-based data plane is real-time data, and our proposed model uses this data for IoT anomaly detection and has superior performance compared to competing models.

The key feature of our solution is that it combines the advantages of the data and control planes. Furthermore, our 1D CNN-based IoT anomaly detection module can be extended by implementing dynamic flow control and mitigation functions with control planes.

We illustrated our idea in experimental network environments. In the future, we can create an SDN network in a real network environment and test our model using the INT data collected from it. Another future work can be that some properties of the network change on the fly, so we can experiment with an adaptive neural network that can continuously learn and provide good performance by adapting to changes in the network.

Additionally, we are working on implementing IoT anomaly detection and mitigation solutions directly at the data plane (in-network) using INT data. We have already implemented a detection solution and mitigation is underway. By implementing IoT anomaly detection and mitigation solutions directly in the data processing path (data plane), detection and mitigation can be faster.

## ACKNOWLEDGEMENT

This research is in part supported by project no. FK_21 138949, provided by the National Research, Development, and Innovation Fund of Hungary.

## REFERENCES


[1] Gereltsetseg ALTANGEREL and Mate TEJFEL. Study on emerging applications on data plane and optimization possibilities. *arXiv preprint arXiv:2204.10186*, 2022.

[2] Perekebode AMANGELE, Martin J. REED, Mays AL-NADAY, Nikolaos THOMOS, and Mateusz NOWAK. Hierarchical machine learning for iot anomaly detection in sdn. In *2019 International Conference on Information Technologies (InfoTech)*, pages 1–4, 2019.

[3] Elisa BERTINO and Nayeem ISLAM. Botnets and internet of things security. *Computer*, 50:76–79, 2017.

[4] Suman Sankar BHUNIA and Mohan GURUSAMY. Dynamic attack detection and mitigation in iot using sdn. In *2017 27th International telecommunication networks and applications conference (ITNAC)*, pages 1–6. IEEE, 2017.

[5] Pat BOSSHART, Dan DALY, Glen GIBB, Martin IZZARD, Nick McKEOWN, Jennifer REXFORD, Cole SCHLESINGER, Dan TALAYCO, Amin VAHDAT, George VARGHESE, and David WALKER. P4: Programming protocol-independent packet processors. *Computer Communication Review*, 44(3):87–95, 2014.

[6] Rodrigo BRAGA, Edjard MOTA, and Alexandre PASSITO. Lightweight ddos flooding attack detection using nox/openflow. In *IEEE Local Computer Network Conference*, pages 408–415, 2010.

[7] Debasrita CHAKRABORTY, Vaasudev NARAYANAN, and Ashish GHOSH. Integration of deep feature extraction and ensemble learning for outlier detection. *Pattern Recognition*, 89:161–171, 2019.

[8] Sharon EZEKIEL, Dinil Mon DIVAKARAN, and Mohan GURUSAMY. Dynamic attack mitigation using SDN. *2017 27th International Telecommunication Networks and Applications Conference, ITNAC 2017*, 2017-Janua:1–6, 2017.

[9] Kostas GIOTIS, Georgios ANDROULIDAKIS, and Vasilis MAGLARIS. Leveraging sdn for efficient anomaly detection and mitigation on legacy networks. In *2014 Third European Workshop on Software Defined Networks*, pages 85–90, 2014.

[10] Jiuxiang GU, Zhenhua WANG, Jason KUEN, Lianyang MA, Amir SHAHROUDY, Bing SHUAI, Ting LIU, Xingxing WANG, Gang WANG, Jianfei CAI, and Tsuhan CHEN. Recent advances in convolutional neural networks. *Pattern Recognition*, 77:354–377, 2018.

[11] Zhiling GUO, Qi CHEN, Guangming WU, Yongwei XU, Ryosuke SHIBASAKI, and Xiaowei SHAO. Village building identification based on Ensemble Convolutional Neural Networks. *Sensors (Switzerland)*, 17(11):1–22, 2017.

[12] Ayyoob HAMZA, Hassan Habibi GHARAKHEILI, Theophilus A BENSON, and Vijay SIVARAMAN. Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity. In *Proceedings of the 2019 ACM Symposium on SDN Research*, pages 36–48, 2019.

[13] MAHMUDUL Hasan, Md MILON Islam, Md ISHRAK Islam Zarif, and MMA HASHEM. Attack and anomaly detection in iot sensors in iot sites using machine learning approaches. *Internet of Things*, 7:100059, 2019.

[14] Daojing HE, Sammy CHAN, Xiejun NI, and Mohsen GUIZANI. Software-defined-networking-enabled traffic anomaly detection and mitigation. *IEEE Internet of Things Journal*, 4(6):1890–1898, 2017.

[15] Alexandre DA SILVEIRA ILHA, Angelo Cardoso LAPOLLI, Jonatas Adilson MARQUES, and Luciano Paschoal GASPARY. Euclid: A Fully In-Network, P4-Based Approach for Real-Time DDoS Attack Detection and Mitigation. *IEEE Transactions on Network and Service Management*, 18(3):3121–3139, 2021.

[16] Qiao KANG, Jiarong XING, and Ang CHEN. Automated attack discovery in data plane systems. *12th USENIX Workshop on Cyber Security Experimentation and Test, CSE T 2019, co-located with USENIX Security 2019*, 2019.

[17] Changhoon KIM, Anirudh SVARAMAN, Naga KATTA, Antonin BAS, Advait DIXIT, Lawrence J WOBKER, and Barefoot NETWORKS. In-band Network Telemetry via Programmable Dataplanes. *Sosr*, (Figure 2):2–3, 2015.

[18] Diederik P. KINGMA and Jimmy Lei BA. Adam: A method for stochastic optimization. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings*, pages 1–15, 2015.

[19] Constantinos KOLIAS, Georgios KAMBOURAKIS, Angelos STAVROU, and Jeffrey VOAS. Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84, 2017.

[20] Nickolaos KORONIOTIS, Nour MOUSTAFA, Elena SITNIKOVA, and Benjamin TURNBULL. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Generation Computer Systems*, 100:779–796, 2019.

[21] Yair MEIDAN, Michael BOHADANA, Yael MATHOV, Yisroel MIRSKY, Asaf SHABTAI, Dominik BREITENBACHER, and Yuval ELOVICI. N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22, 2018.

[22] Francesco MUSUMECI, Ali Can FIDANCI, Francesco PAOLUCCI, Filippo CUGINI, and Massimo TORNATORE. Machine-Learning-Enabled DDoS Attacks Detection in P4 Programmable Net-

works. *Journal of Network and Systems Management*, 30(1), 2022.

[23] Sukhyun NAM, Jiyoon LIM, Jae Hyoung YOO, and James Won Ki HONG. Network Anomaly Detection Based on In-band Network Telemetry with RNN. *2020 IEEE International Conference on Consumer Electronics - Asia, ICCE-Asia 2020*, 2020.

[24] Thien Duc NGUYEN, Samuel MARCHAL, Markus MIETTINEN, Hossein FEREIDOONI, N. ASOKAN, and Ahmad-Reza SADEGHI. DÏot: A federated self-learning anomaly detection system for iot. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 756–767, 2019.

[25] Bruno Astuto A. NUNES, Marc MENDONCA, Xuan-Nam NGUYEN, Katia OBRACZKA, and Thierry TURLETTI. A survey of software-defined networking: Past, present, and future of programmable networks. *IEEE Communications Surveys Tutorials*, 16(3):1617–1634, 2014.

[26] M. B. I REAZ. *Artificial Intelligence Techniques for Advanced Computing Applications*, volume 6. 2013.

[27] Archit SANGHI, Krishna P. KADIYALA, Praveen TAMMANA, and Saurabh JOSHI. Anomaly Detection in Data Plane Systems using Packet Execution Paths. In *SPIN 2021 - Proceedings of the 2021 ACM SIGCOMM Workshop on Secure Programmable network INfrastructure*, number 1, pages 9–15, 2021.

[28] Mohanad SARHAN, Siamak LAYEGHY, Nour MOUSTAFA, and Marius PORTMANN. NetFlow Datasets for Machine Learning-Based Network Intrusion Detection Systems. *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, LNICST*, 371 LNICST(November):117–135, 2021.

[29] Huasong SHAN, Qingyang WANG, and Qiben YAN. Very short intermittent ddos attacks in an unsaturated system. 4 2018.

[30] Valentina TIMČENKO and Slavko GAJIN. Machine learning based network anomaly detection for iot environments. In *ICIST-2018 conference*, 2018.

[31] Enkhtur TSOGBAATAR, Monowar H. BHUYAN, Yuzo TAENAKA, Doudou FALL, Khishigjargal GONCHIGSUMLAA, Erik ELMROTH, and Youki KADOBAYASHI. DeL-IoT: A deep ensemble learning approach to uncover anomalies in IoT. *Internet of Things*, 14(March):100391, 2021.

[32] Nguyen Van TU, Jonghwan HYUN, Ga Yeon KIM, Jae Hyoung YOO, and James Won Ki HONG. INT-Collector: A High-performance Collector for In-band Network Telemetry. *14th International Conference on Network and Service Management*, pages 10–18, 2018.

[33] Ying ZHANG. An adaptive flow counting method for anomaly detection in sdn. In *Proceedings of the Ninth ACM Conference on Emerging Networking Experiments and Technologies*, CoNEXT '13, page 25–30, New York, NY, USA, 2013. Association for Computing Machinery.

Received April 12, 2023, accepted June 16, 2023

## BIOGRAPHIES

**Máté Tejfel** received his B.Sc., M.Sc. and Ph.D. Degree in Computer Science, from ELTE, Budapest Hungary. He is currently working as an Associate Professor in the Department of Programming Languages and Compilers, ELTE. His research interest includes programming languages, correctness check, Software Defined Networks, and network optimization.

**Gereltsetseg Altangerel** received her B.Sc. M.Sc. Degree in Information technology, from Mongolian University of Science and Technology (MUST). She is currently a Ph.D. student in the Department of Programming languages and compilers, ELTE under the supervision of Professor Tejfel Máté. Her research interest includes Software-defined Networks, deeply programmable network, and network optimization.

**Enkhtur Tsogbaatar** received B.Sc. and M.Sc. degrees in Information Network from MUST, Mongolia, in 2011 and 2013, and Ph.D. degree in information science from the Nara Institute of Science and Technology (NAIST), Japan, in 2022, respectively. She is now a Senior Lecturer at the Information Network and Security Department, the School of Information Science and Technology, MUST. Her research interests include machine learning, anomaly detection, IoT security, and information security risk assessment.