Detection and Classification of Botnet Traffic using Deep Learning with Model Explanation

Partha Pratim Kundu, *Member, IEEE,* Tram Truong-Huu, *Senior Member, IEEE,* Ling Chen, Luying Zhou, and Sin G. Teo, *Member, IEEE*

Abstract—Distributed denial-of-service attacks are a kind of malicious attempts among many others that make target services unavailable to legitimate users by using a large number of bots, which send many service requests exceeding the processing capacity of the services. Detection of botnet traffic is therefore critical to maintaining the availability and quality of the services. In contrast, identifying the type of botnet attacks helps system administrators quickly determine which part of the computer and network system is under attack. Current works focus on rule-based detection, which sets rules in the network firewall to drop suspicious traffic that matches the rules. With the emergence of machine learning and deep learning (ML/DL), several preliminary works have been developed to learn botnet traffic behavior and perform detection. However, the performance of existing ML/DL models can be further improved and their decision/prediction are not transparent, making it hard for users to interpret and trust the results. In this work, we develop a novel deep learning model for botnet detection and classification combined with its ability of explaining the decision of the model. We first leverage latent representation of traffic features generated using convolutional neural networks to detect whether a traffic record is generated by a bot then determine the type of bots. We adopt an existing explainable framework to interpret the prediction of the developed deep learning model. We perform extensive experiments with real network traffic as well as synthetic traffic generated by IXIA BreakingPoint System. We compare the developed model with existing models on various performance metrics. The experimental results show that the developed model outperforms the existing machine learning models with an improvement of up to 15% for all performance metrics while providing a clear explanation of the model decision.

Index Terms—Network Security, Botnet Detection and Classification, Deep Learning, Explainable AI.

1 Introduction

The rapid growth of botnet has posed significant security threats to computer users. Victims of botnets, along with smart devices, have substantially grown in number. Hackers attack a computer or a network for illegal activities using untraceable and exceptional coordinated mechanisms. To achieve this goal, generally, a group of hosts at different geographical locations compromised and controlled by a malicious individual or organization launch an attack simultaneously. This makes the attacks hardly traceable back to the origin due to the complexity of the Internet and creates serious threats against legitimate Internet activities. Information leakage, click fraud, and Email spam can be the first step of a serious problem as using these vulnerabilities, a botnet can infect intelligent devices or computers and then launches more severe attacks in networks like (distributed) denial of service (DDoS) attacks [1]. Botnets are emerging threats with billions of hosts worldwide infected. The majority of those hosts are running Microsoft Windows operating systems [2] and [3].

Bots are commonly referred to as software applications running automated tasks over the Internet. A specific group of bots under a command and control (C2 or C&C) server can form a self-propagating, self-organizing, and autonomous network called botnet [2]. The C&C server will remotely control bots using different communication protocols to install worms, Trojan horses, or backdoors on the systems to compromise them. The strategy evolution between botnet's masters and botnet defenders is an ongoing phenomenon and one side try to win the battle over others. The botnet's strength lies in the massive number of bots, which increases the severity of attacks and its master's ability to hide the bots from being detected by security systems. In the current Internet malware epidemic, botnets are one of the main threats as bots can cooperate towards a common malicious purpose and spread over thousands of computers at a very high speed [4]. An example of popular botnets is the Mirai botnet that spreads through Trojans, exploits Internet-of-Things (IoT) devices such as web cameras, closed-circuit television cameras (CCTV) and other devices with low-security measures and involved 100,000 IoT devices [2]. Detection and classification of botnet traffic become critical for network security to protect the systems and stored data.

The research related to the detection of different botnet families has been a trending topic over the last decade [5] and [6]. With the emergence of machine learning and deep learning (ML/DL), several works have proposed to apply machine learning and deep learning to network security problems [7], [8] and botnet detection [9], [10], [11], [12]. The continuous changing of network measurement statistics and botnet behavior characteristics cause existing rule-based approaches to fail. On the other hand, convolutional

- *P. P. Kundu (corresponding author), L. Chen, L. Zhou and S. G. Teo are with Institute for Infocomm Research (I²R), Agency for Science, Technology and Research (A*STAR), Singapore 138632.*
- *T. Truong-Huu is with Singapore Institute of Technology (SIT), Singapore 138683.*

E-mail: kundupp@i2r.a-star.edu.sg, truonghuu.tram@singaporetech.edu.sg, lingc@i2r.a-star.edu.sg, lzhou@i2r.a-star.edu.sg, teosg@i2r.a-star.edu.sg.

neural networks (CNN) consistently showed improved performance in identifying and recognizing complex patterns in computer vision and other fields [13]. Generally, two-dimensional (2D) convolutional neural networks uses 2D image data. But network traffic in raw format is one-dimensional (1D) data. So, in this paper, we develop a 1D Convolutional Neural Network (1DCNN) classifier to leverage the representation learning [13] capability of convolutional neural networks and use it to identify types of botnets traffics and as well as normal traffic. We develop a tool for traffic feature extraction to provide an enriched set of features which are useful for the convolution layers to learn the best latent representation of benign and botnet traffic.

While ML/DL have demonstrated their capabilities in performing complex data analytic tasks, their prediction or decision mechanism is still a black box for researchers and practitioners. The existing works in botnet detection and classification only report their prediction results and do not explain their outcome. Security analysts see those models as a black box without any transparency and evidence to explain why a given traffic record is identified as the generated traffic from a particular type of botnet attack or as the regular or benign traffic. It is generally difficult for an analyst to manually analyze a highly complex black-box model like a deep neural network with millions of parameters and infer the rationale behind its decision making process [14]. As a result, the analysts cannot determine whether they can trust the decisions made by the model. Various explanation approaches have been proposed to mitigate this problem by interpreting predictions generated from the complex machine learning-based models [15], [16], [17], [18]. These methods explain a classification model using additive feature attribution framework, *i.e.*, they identify the input feature set which the classification gave most importance while producing its prediction. By examining these most impacting features, domain experts could easily verify whether the classifier follows the correct logic according to the intrinsic properties of the domain.

In this paper, we adopt one such popular framework named SHAP [16] to the network security problem to explain the prediction of the proposed botnet detection and classification model to earn the trust of its end users, *i.e.*, security analysts. The output of the explanation framework (*i.e.*, the SHAP value of the features) allows us to understand not only the important features of network traffic samples for botnet classification task but also the *impact direction* of their importance, *i.e.*, whether a feature negatively or positively contributes to that decision making. The contributions of this paper are as follows.

1) We develop a novel botnet detection and classification model using 1DCNN, which can learn complex patterns of botnet traffic.
2) We adopt an explainable AI framework to provide explanations about the outcomes of botnet detection and classification models, thus earning the trust of end-users.
3) We carry out extensive experiments to demonstrate the performance of the detection and classification model and assess the explanation quality of the explanatory framework. We use three datasets (two real traffic datasets and a synthetic dataset generated by IXIA Appliance[1]) to generalize the effectiveness of the 1DCNN model and the explanatory framework.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 presents the details of the proposed framework. Section 4 presents the experimental setup and the analysis of the experimental results. We conclude our paper in Section 5.

2 Related Work

Botnet attacks could cause tremendous damage to organizations and governments, especially those who provide critical services to the public. Besides, the episodes could significantly disrupt online activities such as financial transactions, online shopping, chatting, etc. The first botnet was appeared in 2001 to send spam emails. Since then, many have studied botnet behavior and proposed different methods to detect the botnet. These botnet detection methods are commonly applied at the host or network level. At the host-based level, the technique detects botnets by monitoring the host system behavior for any botnet-related suspicious activities. On the other hand, the network-based detection method captures and analyzes network traffic data for identifying hosts infected with botnets. Furthermore, as machine learning and deep learning have shown good detection performance in various domains, many have proposed botnet detection based on the two learning approaches as discussed.

In host-based detection, several works have been proposed as follows. The framework "BotSwat" [19] was proposed to detect infected machines by botnets by monitoring the launched host processes using external parameters. Another framework, "MineSweeper" [9] detects botnets by analyzing all hidden behaviors of the botnets. Cui *et al.* [10] proposed a botnet detection algorithm that uses outbound connections of the processes to the internet, i.e., monitoring the inputs from mouse, keyboard, and other sources. Law *et al.* [11] proposed a botnet detection method by capturing memory and network events from the machines. Subsequently, the authors correlated them to identify suspicious devices. A clustering method [12] is proposed to detect botnets by clustering user behavior executions especially targeting anomalies. Finally, Oulehla *et al.* [20] proposed a mobile botnet detection based on the deep learning approach. The method can detect mobile botnets by classifying mobile traffic data. All of the above discussed host-based detection methods for botnets have some limitations. For example, many advanced botnets could use evasive techniques to escape detection in [10], [19]. The solution [9] is hard to operate in a fully automated manner and partly still needs to involve humans and some scalability issues in handling the large dataset [12].

Host-based detection does not have the global visibility of the network, thus failing to detect botnet traffic where bots are distributed. Network-based detection overcomes this issue by collecting traffic at the network perimeter or

1. IXIA Appliance: https://support.ixiacom.com/sites/default/files/resources/application-note/multi-app-traffic-gen-applib.pdf

different points of the network. The network-based botnet detection is categorized into signature-based, anomaly-based, domain name server (DNS), and machine learning/deep learning-based. Signature-based techniques [21], [22] find botnets by searching and matching the existing known botnet signatures in the network traffic. The anomaly-based methods [7], [23], [24], [25] use anomaly features extracted from the network traffic to identify and detect botnets. The DNS-based approaches [26], [27], [28] could identify botnets by monitoring DNS traffic which the botnet uses DNS to communicate a command-and-control (C & C) server and detecting any anomalies in DNS queries. For the machine learning/deep learning-based approaches, the authors of [29], [30] proposed to use additional network traffic and other related features to train various botnet detection models. The statistical features are extracted from the traffic to train botnet DDoS detection models using supervised learning methods such as SVM, Naïve Bayes, etc. Similarly,the authors of [31], [32], [33], [34], [35] proposed DDoS botnet detection models using different neural network architectures such as Convolutional Neural Network (CNN) [33], [34], [35], Gated Recurrent Unit Neural Network (GRU) [31] and Recurrent Neural Network/Long Short-Term Memory Neural Network (RNN/LSTM) [32], [34]. All of this botnet detection and classification algorithms do not explain their prediction or outcome. Our method produces explanation along with the prediction results to earn the trust of the domain expert.

3 Proposed Framework for Botnet Detection and Classification with Explanation

The proposed botnet detection and classification framework has three distinct modules, namely Feature extraction module, Botnet detection and classification module, *i.e.*, 1DCNN model and Explanation module. The schematic diagram of the proposed framework is shown in Fig. 1. The network traffic is 1-dimensional data. Here, the traffic stream first enters into the feature extraction module. This module extracts the 199 feature values, which include basic features extracted from packet headers and statistical features after performing traffic aggregation into flows and sessions. These 199 feature values are created to form the feature vector of a traffic record, *e.g.*, 3 minutes of a traffic flow having same identification such as source IP, destination IP, etc. These features are then fed into the 1DCNN model for detection and classification. The explanation module identifies the impact of these features using SHAP values [16] on decision making capabilities of the classifier. We provide the details of the component modules in the following subsections.

3.1 Feature Extraction Module

In this paper, we define various features that can be used by various machine learning (ML) algorithms. It is worth mentioning that by carefully selecting features from raw network packets, and from aggregated flows, algorithms can learn to detect and classify various types of botnet traffic in a network. Depending on aggregation criteria, different features can be extracted for real-time detection and classification. We first provide the definitions of traffic flows and sessions in terms of raw traffic or packets and then present the detailed description of the aggregation technique used here.

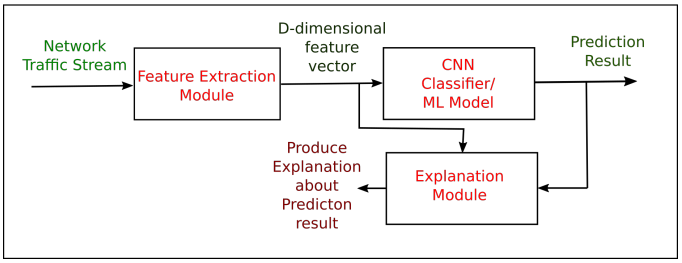


Fig. 1. The overall architecture of the Botnet detection and classification framework.

3.1.1 Packets

A raw traffic stream is consist of a set of packets $P = \{p_1, \ldots, p_{|P|}\}$. Each packet p_i is defined by a set of six tuples $\{sip_i, dip_i, sport_i, dport_i, proto_i, pktsize_i\}$, where sip_i, dip_i, $sport_i$, $dport_i$, $proto_i$ and $pktsize_i$ are source IP address, destination IP address, source port, destination port, protocol used by the packet and size of the packet in bytes, respectively [36].

3.1.2 Flows

A traffic flow is a sequence of packets, localized in time, identified by the 5-tuples namely <source IP address, source port, destination IP address, destination port, protocol> such that the inter-arrival time between two consecutive packets is less than a threshold value, α seconds. If two packets, p_i and p_j share the same 5-tuples but have an inter arrival time greater than the threshold value α, then they are aggregated into two different flows. In this paper, we use $\alpha = 5, 10$ seconds for flow generation.

3.1.3 Sessions

As defined above, flows have an inherent notion of direction. This may create situations where application-level requests or commands are part of one flow, while the response packets are part of a separate flow. To address this situation, we define a traffic session. In a session, the next level of aggregation happens by combining corresponding requests and their reply flows together. Most NetFlow or IPFIX implementations are unidirectional, meaning TCP connections between two hosts results in two flows (*i.e.*, A to B and B to A) that can be stitched back into one bidirectional flow.

To capture more statistical features that could be useful for various botnet attack detection, two or more unidirectional flows could be aggregated into a session that has some common basis to put together. For instance, if a malicious host carries out a scan attack on different victims, the source IP, source port and protocol should be the same for all the generated flows. These flows could then be aggregated into one session. We define a temporal criterion for aggregating flows to create a session. If the inter-arrival time between two consecutive flows is greater than a given threshold value β then these two flows will be part of two separate sessions. The shorter the value of β enables the detection

and classification framework to flag any botnet traffic, *i.e.*, malicious activity quickly.

3.2 Feature Aggregation

In this paper, we extract basic traffic features and produce 4 types of features to feed into the 1DCNN model.

1) **Aggregated features:** These features are computed as a sum, total count, max or min value of a simple feature over a flow or a session using uni- or bi-directional flows.
2) **Temporal features:** These are duration parameters (in seconds) of a flow or a session and their sum, mean and standard deviation.
3) **Statistical features:** These are statistical measures such as mean and standard deviation of aggregated features, over a flow or session of component byte and packets numbers.
4) **Connection-context features:** These features are useful in detecting coordinated attacks by several machines in a compromised network and can be used to recognize multi-flow attacks. "Flow watermarking" identifies sets of flows that have the same/co-related patterns [37] .

We tabulated the description of all the 199 features that we extracted and derived from basic traffic features for the Stratosphere IPS Project [38] and Synthetic data in Table 1 and Table 2 in Appendix A. We encourage interested reader to look into them. We extracted traffic features with different values of β (*e.g.*, 1 and 3 minutes) while varying the value of α (*e.g.*, 5 and 10 seconds).

3.3 Botnet Detection and Classification Module

The application of the convolutional framework in deep neural network has been a great success for computer vision problems. We would like to test this framework for network traffic data. Unlike image data, the nature of network traffic data is one dimensional. So, we use 1DCNN framework. We empirically evaluated various deep neural network architecture on dataset $DS4$ described in Table 2 and tabulated their performance in Table 1. We used batch-normalization, dropout probability with probability value: 0.8 and set number of epochs as 1500, activation function as leaky-relu with leakage factor as 0.1 and loss function as Class-balanced focal loss [39] with parameters $\beta_1 = 0.9$ and $\gamma = 2.1$. We keep all of these parameters same for the architectures mentioned in Table 1 where the first column represents the architecture number, the second column states the number of CNN layers used in the architecture, third column tabulates the details of CNN layers (i.e., filter size $\times$ channel number $\times$ number of filters and ";" separates layer description), the forth column depicts number of Fully Connected (FC) layers used, fifth column tabulates the details of FC layers (input neuron size $\times$ output neuron size; ":" separates layer description) and finally last column shows the F1-score in percentage on the test set, which is mutually exclusive from the training set. The F1-scores of all deep learning architectures mentioned in Table 1 are shown in Fig. 2. From Table 1 and Fig. 2, we observed that

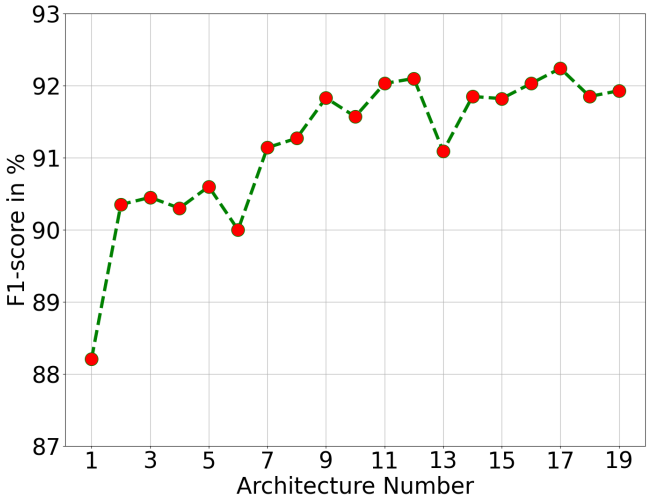


Fig. 2. The F1-score of various deep network architectures.

- If we increase the number of convolution layers the performance of the network improves up to layer 5, after that its performance gain saturates.
- If we increase the number of fully connected (FC) layers more than 3, the performance of the network does not improve at all. Sometime the performance of the network degrades.

We use F1-score as one of the performance metrics to evaluate the developed 1DCNN models.

Keeping all of these observations in mind, in this paper, we use Architecture 12 to detect and classify botnet traffic records from network traffic data. The network traffic stream first go to the feature extraction module. This module produces samples/records for the input layer of 1DCNN model. Each sample has 199 features values extracted from the traffic data. The 1DCNN model we used has 5 convolution (CNN) layers, 3 fully connected (FC) layers, and one `softmax` layer. The detailed schematic of the 1DCNN classification model is shown in Fig. 3. The right side of a convolutional layer has details of the filter size used in it. We used batch-normalization and a Max-pooling layer at the end of each convolutional layer to improve the generalization capability of the network. We put the details of a FC layer inside the box representing it. In our framework, we tested 3 types of loss functions namely Macro-averaged $F1$-score, Focal loss and Class-Balanced Focal Loss. The rationale behind using them is that they are well capable of handling class imbalanced data. We used Adam optimiser [40] for the model training. The description and mathematical formulations of loss functions are presented in Appendix B.

3.4 Explanation Module

As we used a complex and hard-to-interpret machine learning model like deep neural networks for network traffic classification purposes, interpretability is crucial to earn the trust of its end users. Existing work on explaining complex models can be divided into two main categories; global and local explanations depending on whether the framework

TABLE 1
Performance of various 1DCNN architectures

Architure #	# of CNN Layers	Description of CNN layers	# of FC Layers	Description of FC layers	Test set F1-score
1	2	9x1x64;5x1x128	3	6400x512:512x256:256x12	88.21
2	3	13x1x32;9x1x64;5x1x128	3	3200x256:256x12	90.35
3	3	13x1x32;9x1x64;5x1x128	3	3200x200:200x200:200x12	90.45
4	3	13x1x32;9x1x64;5x1x128	3	3200x400:400x400:400x12	90.30
5	3	13x1x32;9x1x64;5x1x128	3	3200x512:512x256:256x12	90.60
6	3	13x1x32;9x1x64;5x1x128	4	3200x1024:1024X512:512x256:256x12	90.00
7	4	17x1x16;13x1x32;9x1x64;5x1x128	3	3200x512:512x256:256x12	91.14
8	4	17x1x32;13x1x32;9x1x64;5x1x128	3	3200x512:512x256:256x12	91.27
9	5	21x1x32;17x1x64;13x1x128;9x1x192;5x1x256	3	3328x512:512x256:256x12	91.83
10	5	17x1x32;13x1x64;9x1x128;5x1x192;3x1x256	3	3328x512:512x256:256x12	91.57
11	5	23x1x32;19x1x64;15x1x128;11x1x192;7x1x256	2	3328x256:256x12	92.03
**12	5	23x1x32;19x1x64;15x1x128;11x1x192;7x1x256	3	3328x512:512x256:256x12	92.10
13	5	23x1x32;19x1x64;15x1x128;11x1x192;7x1x256	4	3328x1024:1024X512:512x256:256x12	91.09
14	5	27x1x32;21x1x64;15x1x128;11x1x192;5x1x256	3	3328x512:512x256:256x12	91.85
15	6	27x1x32;23x1x64;19x1x128;15x1x160;11x1x192;7x1x256	3	1792x512:512x256:256x12	91.82
16	6	23x1x32;19x1x64;15x1x128;11x1x160;7x1x192;5x1x256	3	1792x512:512x256:256x12	92.03
17	7	31x1x32;27x1x64;23x1x128;19x1x160:15x1x192;11x1x224; 7x1x256	3	3328x512:512x256:256x12	92.24
18	7	31x1x32;27x1x64;23x1x128;19x1x160:15x1x192;11x1x224; 7x1x256	2	3328x256:256x12	91.85
19	7	29x1x32;25x1x64;21x1x128;17x1x160:13x1x192;9x1x224; 5x1x256	3	3328x512:512x256:256x12	91.93

tries to describe the model as a whole, *i.e.*, using all of its prediction or tries to explain a specific prediction/output from the model. For both cases the explanation module identify most influenced variables/features responsible model's outcome. Explanation methods may be further divided into two categories: model-specific and model-agnostic (general) explanation methods depending on whether the explanation framework is specific to a particular type of machine learning module.

In this paper, we use a popular model agnostic global explainable AI technique, SHAP [16] to explain predictions of the developed botnet detection and classification model. This framework is based on the Shapley value which is a method originally invented for assigning "payouts" to "players" depending on their contribution towards the total "payout" and builds on concepts from cooperative game theory [41]. In the explanation setting, the features are the "players" and the prediction is the total "payout". SHAP stands for SHapley Additive exPlanations and focuses on the difference between the prediction and the average prediction is perfectly distributed among the features. For instance, if the prediction to be explained is the probability of person P getting symptomatic after getting contacted with COVID-19 virus, the sum of the Shapley values for all features (in this case is the person P's vital parameters, gene mark-ups, etc.) is equal to the difference between this prediction and the mean probability of a person getting symptomatic after getting contacted with COVID-19 virus, where the mean is taken over all persons having contacted with COVID-19 virus. *"all persons"* include all age group, gender, all types of ethnicity, etc.

The advantage of Shapley value is that it has a solid theoretical foundation compared to other explainable AI methods and due to this foundation, it satisfies *Local accuracy*, *Missingness* and *Consistency* properties. The main disadvantage of the Shapley value is that it becomes intractable for high dimensional data as the computational complexity grows exponentially. This has led to approximations like the Kernel SHAP method, the backbone of SHAP framework [16]. This method requires less computational power to obtain a similar level of explanation.

3.4.1 Shapley Value Computation

The exact Shapley value is computed using cooperative game theory. Let there be $\mathcal{D}$ players in total and let $\mathcal{P} \subseteq \mathcal{D}$ be a subset consisting of $|\mathcal{P}|$ players. Here, $\mathcal{D}$ is full feature set of network traffic data. A contribution function $v(\mathcal{P})$ maps subsets of players to the real numbers, called the worth or contribution of coalition $\mathcal{P}$ and describes the total expected sum of "payouts" from the members of $\mathcal{P}$. $v(\mathcal{P})$ can be obtained by this cooperation. According to the Shapley value, a player k gets its fair share from the total "payout" as follows:

$$\begin{aligned}\phi_k(v) &= \phi_k \\ &= \sum_{\mathcal{P}\subseteq\mathcal{D}\setminus\{k\}} \frac{|\mathcal{P}|!(D-|\mathcal{P}|-1)!}{D!}\left(v(\mathcal{P}\cup\{k\})-v(\mathcal{P})\right);\end{aligned} \tag{1}$$

where $k = 1 \ldots D$. So, we can consider ϕ_k is a weighted mean over contribution function of all subsets $\mathcal{P}$ of players not containing player k.

3.4.2 Mapping to Machine Learning Scenarios

Let there be a training set $\{\mathbf{x^i y^i}\}, i = 1, \ldots, n_T$, a predictive model $f(\mathbf{x})$ and target vector $\mathbf{Y}$. n_T is the size of the training set. Our goal is to explain prediction from the model $f(\mathbf{x^s})$ for a specific sample $\mathbf{x} = \mathbf{x^s}$ using Shapley values. We achieve this goal by representing the total "payouts", in this case $f(\mathbf{x} = \mathbf{x^s})$ as follows

$$f(\mathbf{x^s}) = \phi_0 + \sum_{k=1}^{D} \phi_k^s; \phi_0 = E[f(\mathbf{x})]; \tag{2}$$

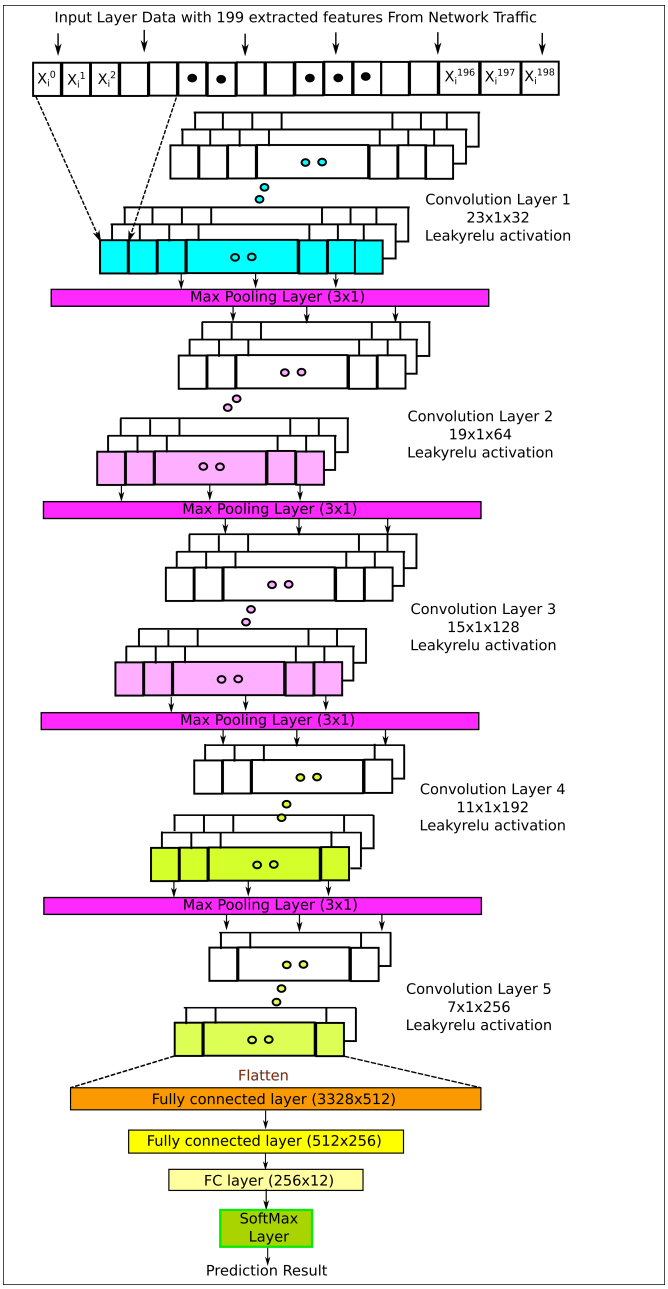


Fig. 3. The Botnet detection and classification aka. 1DCNN framework. i represents sample or record i and j is j^{th} feature. $\mathbf{X}$ is the set of input samples or records.

where $E[f(\mathbf{x})]$ is the global average prediction and ϕ_k^s is the individual "payouts" of feature k for the prediction of a specific sample $x = x^s$. The Shapley values explain the difference between the prediction $\mathbf{y^s} = f(\mathbf{x^s})$ and the global average prediction.

When we consider the contribution for a certain subset $\mathcal{P}^{\int}$, we modify the contribution function as $v(\mathcal{P})$ and compute it as follows

$$v(\mathcal{P}) = E[f(\mathbf{x})|\mathbf{x}_{\mathcal{P}} = \mathbf{x}_{\mathcal{P}^s}]. \quad (3)$$

SHAP framework efficiently compute the Shapley value for individual features using Eqs. (1), (2), and (3) and assuming some assumption in computation of $v(\mathcal{P})$. We refer the reader to [16] for a more detailed description of SHAP framework.

3.4.3 Significance of Shapley Values for Model Transparency

Shapley value-based explanations are used to interpret the complex machine learning model which takes or infer any complex decision automatically without human intervention. Here, "interpret" means to explain or to present a taken decision in human-understandable terms to the user of the model. So it is very important that the decision making logic of the model must be transparent to a human. By highlighting important features and their rough range (which enable the complex model arriving at a decision) in terms of Shapley values make the model transparent and understandable to its users. So the goal of computing Shapley values is not selecting important features that assist the model to take a decision better way rather when a complex model took any decision, the Shapley values' job are to present its complex decision space in a simpler way such that it could be understandable to human. This in turn helps a human to understand the intuition that the complex model follows while arriving at a decision and makes its end user, *i.e.,* a human feel more confident about its decision making process.

4 Experimental Setup and Performance Evaluation of Framework

4.1 Datasets

In order to evaluate the performance of the proposed framework, we used three datasets. The details of these datasets are described below.

4.1.1 Stratosphere IPS Dataset

The Stratosphere IPS Project [38] is continually obtaining malware and normal traffic and store in PCAP files. The dataset includes 11 botnet families containing different attacks that spread over the Internet. Using various combination of α and β for traffic sampling and feature extraction, we have created 4 datasets from the raw traffic (*i.e.,* PCAP files). Each dataset contains traffic samples (which are 1-dimensional feature vectors) to be classified as benign or malicious with specific class of botnet. The class-wise sample distribution of the datasets is presented in Table 2. We have extracted features with different aggregation criteria discussed in Section 3.2. The name and description of 199 features are presented in Table 1 and Table 2 in Appendix A.

4.1.2 Kitsune Dataset

We refer preprocessed Kitsune Surveillance Network Intrusion Datasets [42] to as Kitsune dataset in our paper. The number of features extracted in the Kitsune dataset is different from other two datasets as the feature extraction methods was different from ours and was described in [42]. The class-wise sample distribution of the Kitsune datasets is presented in Table 3.

TABLE 2
Class-wise number of records/samples from the Stratosphere IPS Project dataset

Software type	Software Family	DS1 $\alpha = 5s$ $\beta = 1m$	DS2 $\alpha = 10s$ $\beta = 1m$	DS3 $\alpha = 5s$ $\beta = 3m$	DS4 $\alpha = 10s$ $\beta = 3m$
Benign	Normal	1299434	1109930	1379736	1131939
Malware	Andromeda	216452	215234	216452	215234
	Barys	101333	83612	101333	83612
	Emotet	442501	423762	442501	204075
	Dridex	87800	84822	87800	84822
	Artemis	73069	71507	73069	71507
	Miuref	131919	123080	131919	123080
	Necurs	27025	25533	27025	25533
	Sality	724718	310009	825596	79628
	Trickbot	749324	753586	779471	724446
	Ursnif	11269	10924	11269	10924
	Wannacry	41266	40069	41266	40069

TABLE 3
Class-wise number of records/samples from Kitsune dataset

Software Type	Software Family	Kitsune
Benign	Normal	16166317
Malware	Active_Wiretap	923216
	ARP_MitM	1145272
	Fuzzing	432783
	OS_Scan	65700
	SSDP_Flood	1439604
	SSL_Renegotiation	92652
	SYN_DoS	7038
	Video_Injection	102499
	Mirai	642516

4.1.3 Synthetic Dataset

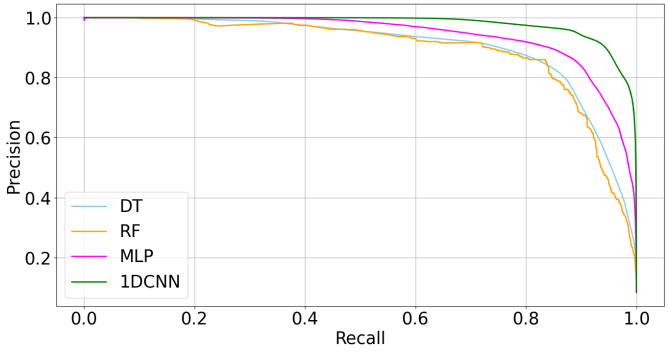
A synthetic dataset was generated using IXIA PerfectStorm One: BreakingPoint System (BPS) by simulating enterprises, government agencies traffic, and general Internet traffic using various protocols with 300 nodes which includes switches, routers and end hosts. It generates real-time attacking traffic that exploits and exposes security vulnerabilities on network infrastructure components. All end hosts generate normal traffic during the normal working condition. When a simulated attack is launched, 100 end hosts were set as the compromised hosts and generate Mirai botnet traffic. We used the IP address of those end hosts to perform traffic labeling for the training purposes. The class-wise sample distribution of the Synthetic datasets is mentioned in Table 4. The feature aggregation scheme and description is the same as Stratosphere IPS Project dataset. The parameters used as $\alpha = 10s$ and $\beta = 1$min.

4.2 Analysis of Experimental Results

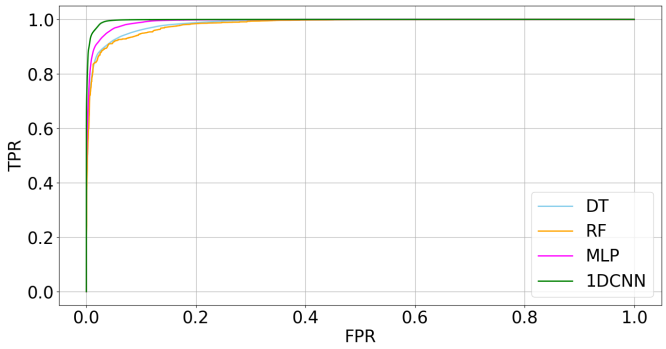
In this section, we compare the performance of the proposed botnet detection and classification module, i.e., 1DCNN models with other well-known ML models. We divide each dataset into two mutually exclusive sets, namely the training set and test set. The training set contains 80% of the samples that are in the dataset. The test set contains the remaining 20%. We used decision tree (DT), Random forest (RF) and Multi-layer perception (MLP) models for comparison. One of the similar state-of-the-art technique is described in [43] where DT and MLP classifiers used for classification task. To make a comparison with the study, we included these 3 classifiers in our paper. The datasets used

TABLE 4
Class-wise number of records/samples from Synthetic dataset

Software Type	Software Family	Synthetic
Benign	Normal	482524
Malware	DNS Flood	416
	HTTP Flood	78375
	UDP Flood	84983
	UDP-PF	227
	VSEQF	16208



(a) Precision-Recall Curve



(b) AUC-ROC Curve

Fig. 4. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS2$ of various classifiers. Legend showing the name of the classifiers is used.

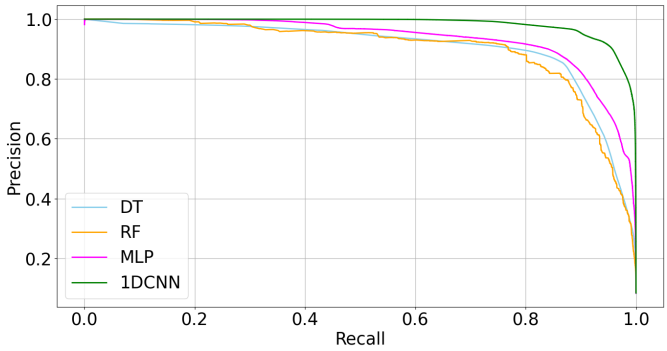
in those works have botnet traffic classes that are different from ours and the feature extraction method was also not same as ours. So we do not follow the exact implementation of the methods described in the paper [43].

The performance results on the test sets of the 4 datasets described in Table 2 are presented in Table 5. We evaluated the performance of ML and DL models using 5 evaluation metrics namely Accuracy, Precision Score, F1-score, Area Under the Curve of Receiver Operating Characteristic (AUC-ROC) and Area Under the Curve of Precision-Recall Curve (AUC-PRC) and tabulated their scores from the third column to the seventh column, respectively, in Table 5. The first and second columns contain dataset name and the classifier name, respectively. The 1DCNN classifier model performs the best in all evaluation metrics and on all datasets. The improvement is by up to 15%.

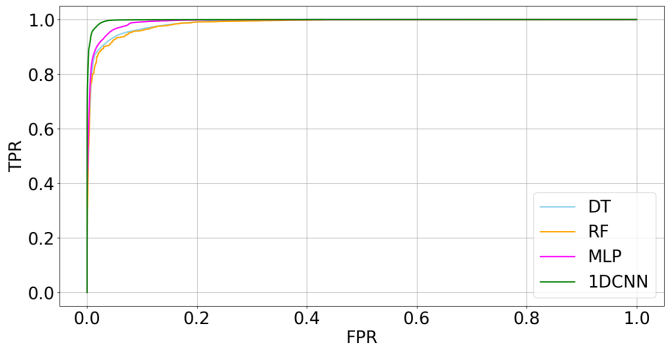
From Table 5, it can be noted that the performance of 1DCNN on data ($DS1$ and $DS2$) with shorter session duration with $\beta = 1m$ is similar on the data with $\beta = 3m$. This means that our framework is able to identify malicious

TABLE 5
Performance comparison of 1DCNN classifier model with other ML models with four Stratosphere IPS Project datasets. All evaluation metrics are in percentage.

Data Set Name	Classifier Models	Evaluation Metrics				
		Accuracy	Precision	F1-score	AUC-ROC	AUC-PRC
DS1 $\alpha = 5$s $\beta = 1$m	DT	85.00	89.96	84.01	96.90	90.32
	MLP	87.86	94.16	87.45	98.57	94.17
	RF	85.20	89.95	84.21	97.27	90.37
	1DCNN	**92.23**	**98.00**	**92.14**	**99.56**	**97.96**
DS2 $\alpha = 10$s $\beta = 1$m	DT	87.01	90.15	86.13	97.12	90.54
	MLP	88.00	93.51	87.00	98.63	93.52
	RF	86.54	89.99	83.55	97.88	90.01
	1DCNN	**92.98**	**98.34**	**92.88**	**99.62**	**98.34**
DS3 $\alpha = 5$s $\beta = 3$m	DT	86.56	90.47	85.56	96.77	90.83
	MLP	88.54	94.73	88.24	98.60	94.93
	RF	85.16	90.01	83.17	97.34	90.00
	1DCNN	**93.30**	**98.50**	**93.28**	**99.63**	**98.44**
DS4 $\alpha = 10$s $\beta = 3$m	DT	86.39	88.66	84.38	97.04	89.06
	MLP	87.54	94.23	87.98	98.76	94.55
	RF	87.46	92.36	85.56	98.58	92.38
	1DCNN	**92.08**	**97.50**	**92.10**	**99.50**	**97.48**

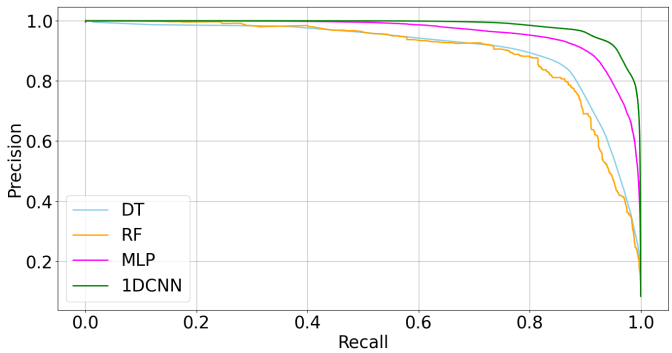


(a) Precision-Recall Curve

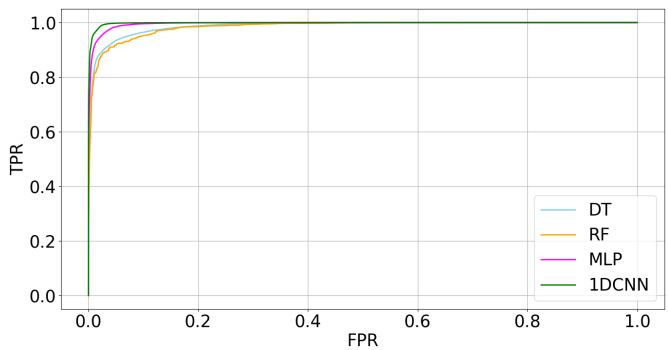


(b) AUC-ROC Curve

Fig. 5. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS2$ of various classifiers. Legend showing the name of the classifiers is used.



(a) Precision-Recall Curve

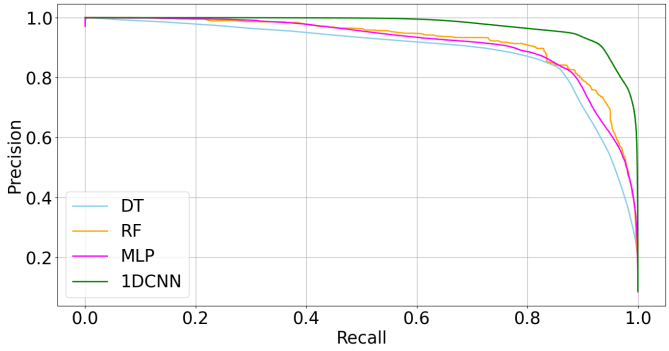


(b) AUC-ROC Curve

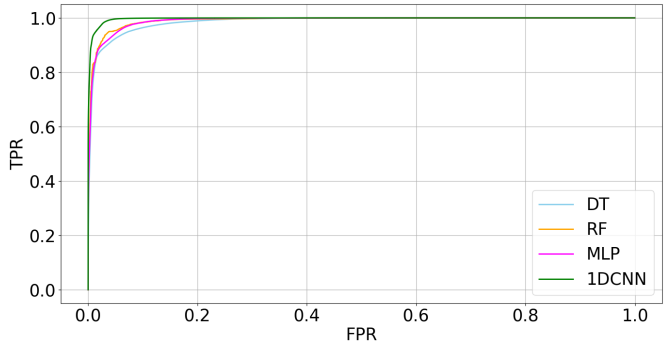
Fig. 6. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS3$ of various classifiers. Legend showing the name of the classifiers is used.

behaviour within $1m$ with a very high accuracy. The detection time within $1m$ is acceptable to many end users. This early detection is very useful in view of system protection. One related observation from Table 5 is that the combination of shorter α's value and longer β's value produces better performance for the 1DCNN model.

The Precision-Recall Curves and AUC-ROC Curves on datasets $DS1$, $DS2$, $DS3$ and $DS4$ of various classifiers are shown in Fig. 4, Fig. 5, Fig. 6, and Fig. 7, respectively. The experimental results show that the 1DCNN classifier performs the best compared to all other classifiers. This performance improvement is more prominently visible in Precision-Recall Curves (Figs. 4a - 7a). The closer looks at AUC-ROC curves of the 1DCNN model on all four datasets reveal that they have over 91% TPR (Recall) value at 0.1% FPR. The best performance from method I and II

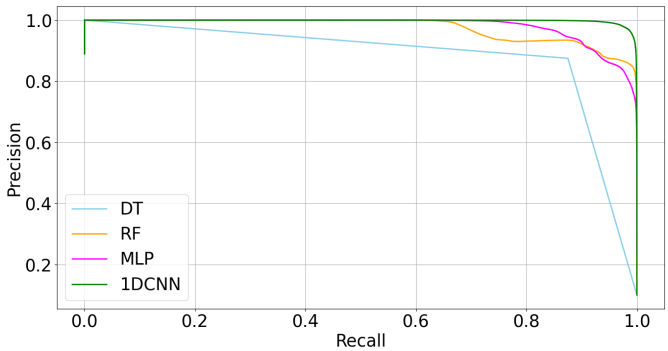


(a) Precision-Recall Curve

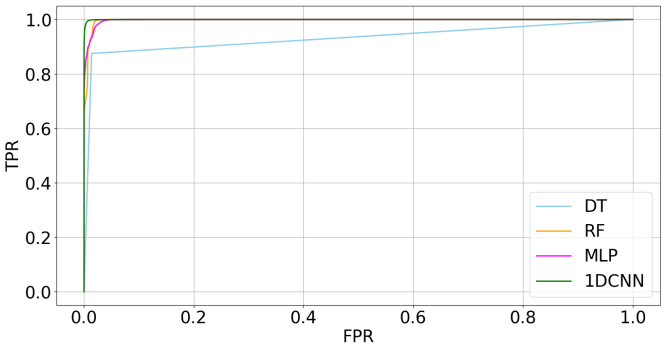


(b) AUC-ROC Curve

Fig. 7. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS4$ of various classifiers. Legend showing the name of the classifiers is used.



(a) Precision-Recall Curve



(b) AUC-ROC Curve

Fig. 8. The Precision-Recall Curve and AUC-ROC Curve on Kitsune dataset of various classifiers. Legend showing the name of the classifiers is used. Clearly 1DCNN classifier outperformed all the classifiers.

described in the paper [43] is 89.9% TPR at 0.1% FPR. This demonstrates that the 1DCNN model has higher TPR value compared to the existing work. This fact establishes the point that our framework beats one of the state-of-the-art method in terms of the classification performance.

We measure the performance of ML and DL models using five performance metrics on Kitsune and Synthetic datasets. The performance results on the test sets are presented in Table 6. The test was created using 20% of the total data. Its samples do not have any overlap with training data. The structure of this table is the same as that of Table 5. We put the experimental results of these two datasets in separate table because Kitsune data does not have the same features as the Stratosphere IPS Project data as mentioned previously. Furthermore, the synthetic data is not real, i.e., we generated it in our lab using IXIA PerfectStorm One machine. The 1DCNN model outperforms all other machine learning models with respect to 5 evaluation metrics on these two datasets. For the synthetic data, the 1DCNN model performs nearly at 100% in some evaluation metrics. One reason could be that IXIA PerfectStorm One machine could not able to simulate all realistic network traffic scenarios. The Precision-Recall Curve and AUC-ROC Curve generated from experiments on the Kitsune and Synthetic datasets with various classifiers are shown in Fig. 8 and Fig. 9, respectively. The results show that the 1DCNN model also performs better than other classifiers.

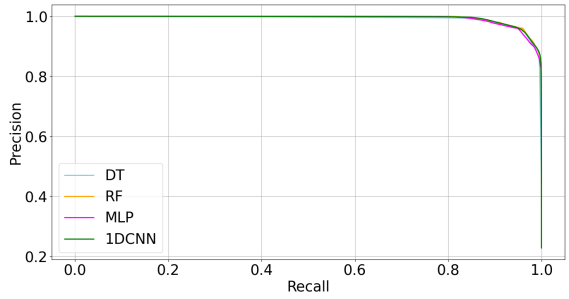
4.3 Explanation of Model Decision

As we discussed in Section 3.4, SHAP generates explanation of predictions of a classifier by identifying the most important features based on a feature attribution framework and Shapley values. In other words, Shapley values tell us on which features the classifier mostly depends on when performing the classification task. To compute Shapley values according to Eq. (2), we need to compute $E[f(\mathbf{x})]$ over a set of samples $\mathbf{x}$. Due to the limit of computational resources (we trained and tested the developed models on a customized GPU desktop with 2 graphic cards Nvidia GTX 2080, 11 GB), we could only be able to use maximum 30 samples to compute the average global prediction, $E[f(\mathbf{x})]$. As we would like to examine how the 1DCNN classifier predicts each class individually, we are interested on those samples which are correctly classified by the model. We did a k-means clustering [44] with cluster number $k = 30$ on these correctly classified traffic records for each class. We use these 30 cluster means as $\mathbf{x}$ to $E[f(\mathbf{x})]$ for the SHAP framework.

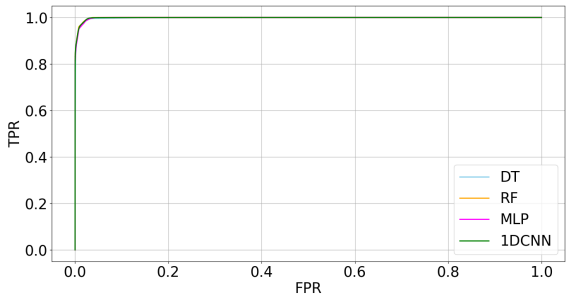
For this experiment, we used dataset $DS2$ created from the raw traffic of Stratosphere IPS. We choose this one for explanation as it is the best performing dataset among the datasets with $\beta = 1m$. We randomly select 500 samples/records from the test set for each class to produce explanations in form of Shapley values. We show the top 20 class-wise important features by sorting the sum of magnitudes of Shapley values over these samples for $DS2$ in Fig. 12 and Fig. 13. These plots are called summary plots.

TABLE 6
Performance comparison of 1DCNN model with other ML models with Kitsune and Synthetic datasets. All evaluation metrics are in percentage.

Data Set Name	Classifier Model	Evaluation Metrics				
		Accuracy	Precision	F1-score	AUC-ROC	AUC-PRC
Kitsune	DT	87.51	77.83	87.50	87.35	88.13
	MLP	91.20	98.10	91.01	99.50	98.09
	RF	92.20	97.35	92.10	99.83	97.43
	1DCNN	**97.90**	**99.85**	**97.72**	**99.90**	**99.80**
Synthetic	DT	96.71	99.51	96.11	99.52	99.49
	MLP	95.84	99.48	95.83	99.48	99.37
	RF	96.95	99.39	95.97	99.51	99.41
	1DCNN	**97.05**	**99.85**	**97.00**	**99.89**	**99.79**



(a) Precision-Recall Curve



(b) AUC-ROC Curve

Fig. 9. The Precision-Recall Curve and AUC-ROC Curve on Synthetic dataset of various classifiers. Legend showing the name of the classifiers is used.

It is worth mentioning that the actual class and predicted class by the 1DCNN model of these selected 500 samples are the same. The Shapley values indicate the impacts of each feature on the model's output; in this case, predicting a particular class. The color shown in the plots represent the relative magnitude of feature values: the red color indicates high values whereas the blue color indicates low values. As we are using 500 samples, we have 500 Shapley values for each feature. In the plots, the straight line above zero indicates the neutral line, representing the $E[f(\mathbf{x})]$. For instance, if all the Shapley values for a feature fall on this line, that feature does not have any impact on the model outcome. The Shapley values that fall right side of this line have a positive impact while those values in the left side of the neutral line have negative impact on the model outcome. Here, positive impact of a feature means the specified range of values of the feature leads a classifier to classify the samples from the class in concern correctly, *i.e.*, the specified range helps traffic records fall within the class boundary of the class in concern. The negative impact means the vice-versa.

Based on the Shapley values obtained from our experiments on the network traffic datasets, we draw several important observations as follows.

1) The high session duration (*sess_dur*) with $\beta = 1$ minute has a negative impact on the model decision for Normal class (Fig. 12a): the normal traffic samples with long session duration tend to be predicted as botnet traffic. Whereas the same has a positive impact for botnet classes specially for Artemis (Fig. 12f), Sality (Fig. 13c), Trickbot (Fig. 13d), Ursnif (Fig. 13e) and Wannacry (Fig. 13f).
2) The high value of *sport* has a negative impact for Normal class (Fig. 12a) whereas the same has a positive impact for most of the botnet classes, e.g., Andromeda (Fig. 12b), Artemis (Fig. 12f), Miuref (Fig. 13a), Wannacry (Fig. 13f), etc. One important observation is that the high value of *sport* has a negative impact in classification for Necurs (Fig. 13b) and Trickbot (Fig. 13d) classes. The 1DCNN model predicts these two botnet classes as a normal traffic (i.e., false negative) if the value of *sport* is high.
3) The byte rate related features, *e.g.*, *sum_bytes_rate*, *max_bytes_rate*, etc. and packet rate related features, *e.g.*, *max_tot_pktsrate*, *min_tot_pktsrate*, etc. have high positive impact for classification among various botnets (Fig. 12 and Fig. 13) compared to Normal class. The Andromeda, Necurs and Sality class data have very high dependence on these type of features for correct classification.
4) The high value of *dport* has a negative impact on the model decision of Normal traffic. The Dridex and Emotet botnet also have high sensitivity to this feature. Thus. they are unique among other botnet classes. The high values of *dport* make the model correctly predict these class as shown in Fig. 12d and Fig. 12e, respectively.
5) We have extracted forward and backward direction features separately. None of them appeared in the top 20-feature list with respect to SHAP values. So, we explore the data and found that 99.99% samples have zero value in those features. Thus, for further study on the botnet detection and classifi-

cation problems, we may not need to extract this feature from raw traffic. This information is helpful for online implementation of this framework as feature extraction from every single packet is a time-consuming task given the high speed of networks.

6) The header-related features *mean_tot_header_len*, *min_tot_header_len*, *sum_tot_header_len*, etc. have frequently appeared in top 20 feature list for most summary plots in Figs. 12 and 13. According to botnet classification literature [9], [10], [11], [12], the packet header plays a significant role for classifying the botnets.

These observations highlight the fact that the model outcomes have parity with the domain knowledge. These supporting facts are also helpful for earning the trust from end users such as security experts. It is worth mentioning that the impact direction information of relative features on the model decision are not available when using the feature importance functionality of Random Forest (RF) and Decision Tree (DT), which select features that discriminate between class boundaries. However, RF/DT frameworks fail to highlight relative effects of features on class boundary formation. We also obtained similar summary plots on the SHAP values on $DS1$, $DS3$, $DS4$ and synthetic datasets. Due to space limit, we do not put them in this paper.

We would like to mention that the interpretation (refer to Fig. 12 and Fig. 13) generated here using the 500 samples from the test set. The deep learning model whose decision we are explaining did not see this set of samples during its training phase. This point establishes the fact that SHAP framework is quite capable of producing explanation of model's prediction or decision making process using previously unseen data. But these unseen samples belong to the classes which the model have seen during its training phase. So, how the SHAP framework generates explanations due to the samples from previously unseen classes would be an interesting problem for future investigation.

4.4 Ablation Study

We did an ablation study using Architecture 12 and 17 by varying three loss functions as discussed in Section 3.3 on dataset $DS4$. We choose these two architecture as they are the best performing among the others as shown in Fig. 2. The class-wise and overall F1-score on test data of F1-loss, Focal Loss (FL), and Class-balanced Focal Loss ($CBFL$) functions are tabulated in the third, fourth and fifth column of Table 7, respectively. The first two columns of this table contain architecture number that is coming from Table 1 and class name of the data, respectively.

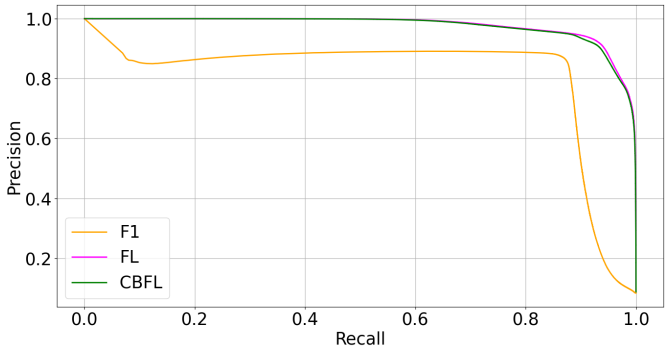
Even though FL has the best performance in terms of overall F1-score, the F1-score with respect to Ursnif class is zero for both architectures. Ursnif is the class with the smallest number samples. This means that the models using FL fail to detect a single sample from this class of botnet. From cybersequrity point of view, this is not acceptable. In case of F1-loss, the F1-score is low for the overall as well as other classes with respect to other losses. Only the architectures trained with $CBFL$ are able to detect samples from all the class with relatively high F1-score. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS4$ with respect to these three loss functions are shown in Fig. 10 and Fig. 11 for both architectures, respectively. The performance of FL and $CBFL$ are almost similar and are far better than F1-loss. We performed the same experiments with $DS1$, $DS2$, $DS3$ and Kitsune dataset and we obtained similar results.

TABLE 7
F1-score comparison of 1DCNN model varying loss functions on dataset $DS4$. All scores are in percentage

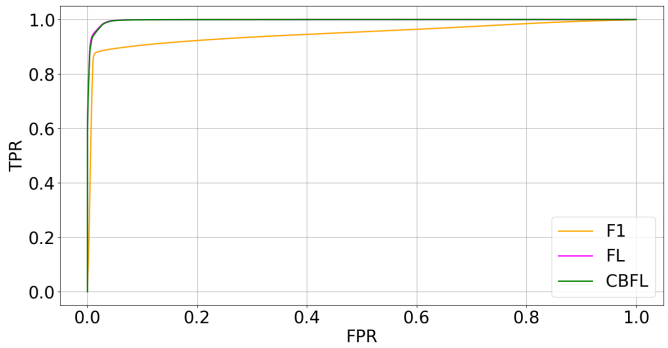
Architecture Number	Class Name	Loss Name		
		F1-loss	FL	CBFL
12	**Normal**	87.27	93.35	93.13
	Andromeda	92.31	94.04	93.68
	Barys	98.16	99.00	98.97
	Emotet	81.43	90.97	90.87
	Dridex	78.19	97.24	96.93
	Artemis	66.78	76.64	59.92
	Miuref	74.19	84.27	84.34
	Necurs	60.13	87.98	84.29
	Sality	85.01	96.52	96.08
	Trickbot	92.63	94.32	94.16
	Ursnif	24.02	00.00	39.63
	Wannacry	82.03	96.35	95.83
	Overall	86.92	92.73	92.10
17	**Normal**	88.34	93.27	93.45
	Andromeda	83.67	94.12	94.17
	Barys	97.92	98.87	99.33
	Emotet	81.77	90.90	90.71
	Dridex	90.69	96.72	95.92
	Artemis	70.81	76.62	59.07
	Miuref	79.38	83.62	84.98
	Necurs	78.39	87.37	84.51
	Sality	84.55	96.42	96.19
	Trickbot	92.78	94.18	94.25
	Ursnif	20.58	00.00	41.80
	Wannacry	88.23	95.84	97.02
	Overall	87.50	92.61	92.24

5 Conclusion

Though botnet detection and classification is a well studied problem, we are motivated by the lack of efforts in gaining the trust of end-users through interpretation of model decision. In this paper, we not only designed a novel botnet detection and classification framework but also provided evidences and explanation to trust its outcome. We carried an exhaustive study of various 1DCNN architectures to determine the best architecture along with the most suitable loss function maximizing the performance of the models. We adopted the SHAP framework with an aim to interpret model decision based on the impact direction and significance of features. We evaluated the performance of the proposed framework using three datasets including two real network traffic datasets and one synthetic dataset created using IXIA BreakingPoint System. The experimental results show that the developed 1DCNN model outperforms other state-of-the-art ML models with an improvement of up to 15% in all performance metrics. The experimental results obtained from the SHAP framework also provides clear explanation of the 1DCNN model's outcome through SHAP's additive feature attribution. These explanations reveal various intrinsic and important characteristics of normal traffic and various botnet traffic families. It also validates the fact

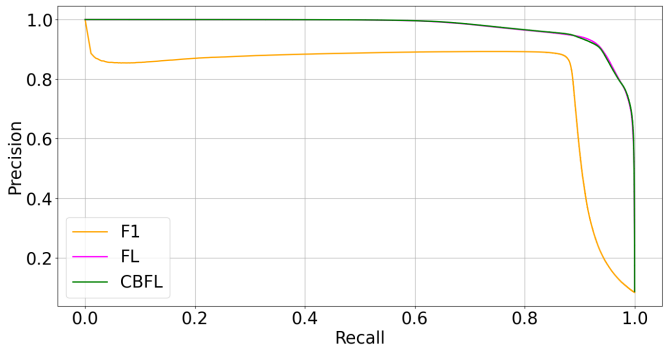


(a) Precision-Recall Curve of Architecture 12

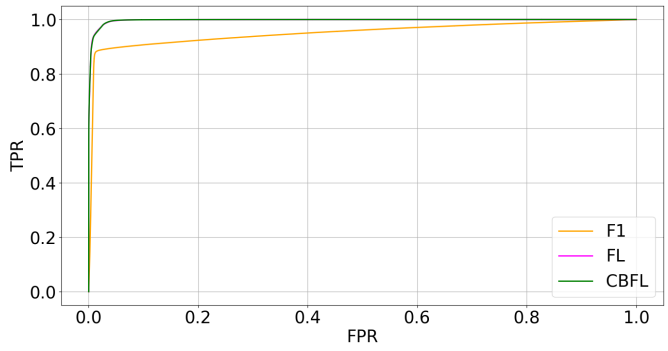


(b) AUC-ROC Curve of Architecture 12

Fig. 10. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS4$ of various loss functions of Architecture 12. Legend shows the names of the losses used.



(a) Precision-Recall Curve of Architecture 17



(b) AUC-ROC Curve of Architecture 17

Fig. 11. The Precision-Recall Curve and AUC-ROC Curve on dataset $DS4$ of various loss functions of Architecture 17. Legend shows the names of the losses used.

that the model predictions have parity with the domain knowledge which increase model trustworthiness.

REFERENCES

[1] K. Ono, I. Kawaishi, and T. Kamon, "Trend of botnet activities," in *2007 41st Annual IEEE International Carnahan Conference on Security Technology*, 2007, pp. 243–249.

[2] W. N. H. Ibrahim, S. Anuar, A. Selamat, O. Krejcar, R. González Crespo, E. Herrera-Viedma, and H. Fujita, "Multilayer framework for botnet detection using machine learning algorithms," *IEEE Access*, vol. 9, pp. 48 753–48 768, 2021.

[3] Wikipedia. Botnet. [Online]. Available: https://en.wikipedia.org/wiki/Botnet

[4] J. Govil, "Examining the criminology of bot zoo," in *2007 6th International Conference on Information, Communications Signal Processing*, 2007, pp. 1–6.

[5] M. Eslahi, R. Salleh, and N. B. Anuar, "Bots and botnets: An overview of characteristics, detection and challenges," in *2012 IEEE International Conference on Control System, Computing and Engineering*, 2012, pp. 349–354.

[6] J. Liu, Y. Xiao, K. Ghaboosi, H. Deng, and J. Zhang, "Botnet: Classification, Attacks, Detection, Tracing, and Preventive Measures," *EURASIP J. Wirel. Commun. Netw.*, vol. 2009, February 2009.

[7] T. Truong-Huu, N. Dheenadhayalan, P. P Kundu, V. Ramnath, J. Liao, S. G. Teo, and S. Praveen Kadiyala, "An Empirical Study on Unsupervised Network Anomaly Detection Using Generative Adversarial Networks," in *1st Security and Privacy on Artificial Intelligent Workshop (SPAI'20)*, Taipei, Taiwan, Oct. 2020.

[8] T.-D. Pham, T.-L. Ho, T. Truong-Huu, T.-D. Cao, and H.-L. Truong, "MAppGraph: Mobile-App Classification on Encrypted Network Traffic using Deep Graph Convolution Neural Networks," in *Annual Computer Security Applications Conference (ACSAC 2021)*, Virtual Conference, December 2021.

[9] D. Brumley, C. Hartwig, Z. Liang, J. Newsome, D. Song, and H. Yin, "Automatically identifying trigger-based behavior in malware," in *Botnet Detection*. Springer, 2008, pp. 65–88.

[10] W. Cui, R. H. Katz, and W.-t. Tan, "Binder: An extrusion-based break-in detector for personal computers," in *USENIX Annual Technical Conference, General Track*, 2005, pp. 363–366.

[11] F. Y. Law, K.-P. Chow, P. K. Lai, and K. Hayson, "A host-based approach to botnet investigation?" in *International Conference on Digital Forensics and Cyber Crime*. Springer, 2009, pp. 161–170.

[12] H. Lamba, T. J. Glazier, J. Cámara, B. Schmerl, D. Garlan, and J. Pfeffer, "Model-based cluster analysis for identifying suspicious activity sequences in software," in *Proceedings of the 3rd ACM on International Workshop on Security And Privacy Analytics*, 2017, pp. 17–22.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, p. 84–90, May 2017.

[14] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Comput. Surv.*, vol. 51, no. 5, Aug. 2018.

[15] M. T. Ribeiro, S. Singh, and C. Guestrin, ""why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1135–1144.

[16] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

[17] H. Lakkaraju, S. H. Bach, and J. Leskovec, "Interpretable decision sets: A joint framework for description and prediction," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '16. New York, NY, USA: Association for Computing Machinery, 2016, p. 1675–1684.

[18] J. Kazemitabar, A. Amini, A. Bloniarz, and A. S. Talwalkar, "Variable importance using decision trees," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017.

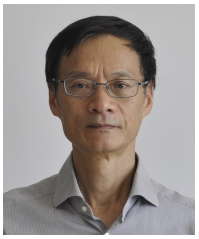
[19] E. Stinson and J. C. Mitchell, "Characterizing bots: Remote control behavior," in *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 2007, pp. 89–108.

[20] M. Oulehla, Z. K. Oplatková, and D. Malanik, "Detection of mobile botnets using neural networks," in *2016 Future Technologies Conference (FTC)*. IEEE, 2016, pp. 1324–1326.
[21] SnortIDS. [Online]. Available: http://www.snort.org
[22] M. Roesch *et al.*, "Snort: Lightweight intrusion detection for networks." in *Lisa*, vol. 99, no. 1, 1999, pp. 229–238.
[23] M. Yamada, M. Morinaga, Y. Unno, S. Torii, and M. Takenaka, "Rat-based malicious activities detection on enterprise internal networks," in *2015 10th International Conference for Internet Technology and Secured Transactions (ICITST)*. IEEE, 2015, pp. 321–325.
[24] D. Jiang and K. Omote, "An approach to detect remote access trojan in the early stage of communication," in *2015 IEEE 29th international conference on advanced information networking and applications*. IEEE, 2015, pp. 706–713.
[25] J. Liao, S. G. Teo, P. P Kundu, and T. Truong-Huu, "ENAD: An Ensemble Framework for Unsupervised Network Anomaly Detection," in *Proc. IEEE CSR 2021*, Virtual Conference, July 2021.
[26] A. Ramachandran, N. Feamster, and D. Dagon, "Detecting botnet membership with dnsbl counterintelligence," in *Botnet Detection*. Springer, 2008, pp. 131–142.
[27] H. R. Zeidanloo and A. B. A. Manaf, "Botnet detection by monitoring similar communication patterns," *arXiv preprint arXiv:1004.1232*, 2010.
[28] D. Mahjoub, "Monitoring a fast flux botnet using recursive and passive dns: A case study," in *2013 APWG eCrime Researchers Summit*. IEEE, 2013, pp. 1–9.
[29] D. Zammit, "A machine learning based approach for intrusion prevention using honeypot interaction patterns as training data," *University of Malta*, pp. 1–55, 2016.
[30] R. Doshi, N. Apthorpe, and N. Feamster, "Machine learning ddos detection for consumer internet of things devices," in *2018 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2018, pp. 29–35.
[31] X. Yuan, C. Li, and X. Li, "Deepdefense: identifying ddos attack via deep learning," in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2017, pp. 1–8.
[32] Y. Meidan, M. Bohadana, Y. Mathov, Y. Mirsky, A. Shabtai, D. Breitenbacher, and Y. Elovici, "N-baiot-network-based detection of iot botnet attacks using deep autoencoders," *IEEE Pervasive Computing*, vol. 17, no. 3, pp. 12–22, 2018.
[33] S.-C. Chen, Y.-R. Chen, and W.-G. Tzeng, "Effective botnet detection through neural networks on convolutional features," in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 2018, pp. 372–378.
[34] B. Nugraha, A. Nambiar, and T. Bauschert, "Performance evaluation of botnet detection using deep learning techniques," in *2020 11th International Conference on Network of the Future (NoF)*. IEEE, 2020, pp. 141–149.
[35] S. Hosseini, A. E. Nezhad, and H. Seilani, "Botnet detection using negative selection algorithm, convolution neural network and classification methods," *Evolving Systems*, pp. 1–15, 2021.
[36] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 712–717.
[37] N. Moustafa, "Designing an online and reliable statistical anomaly detection framework for dealing with large high-speed network traffic," Ph.D. dissertation, 10 2017.
[38] S. R. Laboratory. Labeled dataset with botnet, normal and background traffic. [Online]. Available: https://www.stratosphereips.org/
[39] Y. Cui, M. Jia, T.-Y. Lin, Y. Song, and S. Belongie, "Class-balanced loss based on effective number of samples," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9260–9269.
[40] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations (ICLR 2015)*, San Diego, CA, USA, May 2015.
[41] L. S. Shapley, *A Value for N-Person Games*. Santa Monica, CA: RAND Corporation, 1953.
[42] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: An ensemble of autoencoders for online network intrusion detection," in *Network and Distributed Systems Security (NDSS) Symposium*, 2018, appears in Network and Distributed Systems Security Symposium (NDSS) 2018; null ; Conference date: 18-02-2018 Through 21-02-2018.
[43] F. K. Wai, Z. Lilei, W. K. Wai, S. Le, and V. L. L. Thing, "Automated botnet traffic detection via machine learning," in *TENCON 2018 - 2018 IEEE Region 10 Conference*, 2018, pp. 0038–0043.
[44] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

Partha Pratim Kundu is a Scientist at Institute for Infocomm Research (I^2R), Agency for Science, Technology and Research (A*STAR), Singapore. Prior to joining I^2R, he served as a post-doctoral researcher in the Department of Computer Science and Engineering, Nanyang Technological University (NTU), Singapore and in the Department of Computer Science and Electrical Engineering, Syracuse University, USA. He received his Ph. D. from Indian Statistical Institute, Kolkata in Machine learning (AI) and Master degree in computer science and engineering with gold medal. He is an author of many top rated journal/conference papers. His current research interest includes deep learning, explainable AI and applications of machine learning algorithms in cyber security domain. He is a member of the IEEE since 2009.

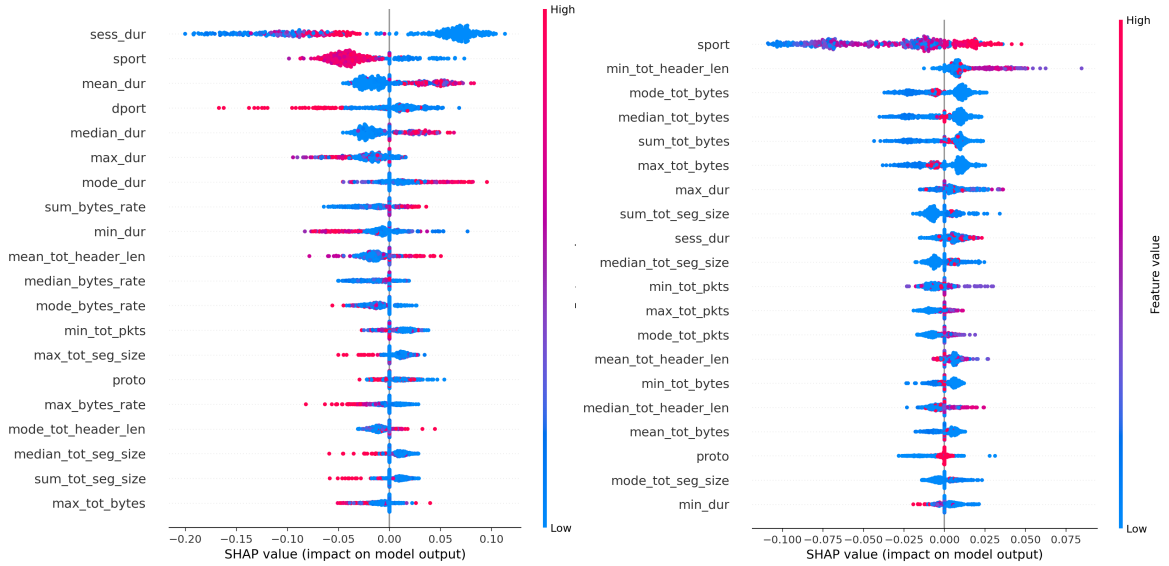
Tram Truong-Huu is an Assistant Professor at the Singapore Institute of Technology (SIT), Infocomm Technology (ICT) Cluster. He is currently holding a joint appointment with Agency for Science, Technology and Research (A*STAR), Singapore, where he has worked as a computer scientist at Institute for Infocomm Research (I^2R) since May 2019. He received the Ph.D. degree in computer science from the University of Nice - Sophia Antipolis (now Côte d'Azur University), France in December 2010. From January 2011 to June 2012, he held a post-doctoral fellowship at the French National Center for Scientific Research (CNRS), France. He worked at the National University of Singapore as a research fellow from July 2012, then senior research fellow from January 2017. His research interests include software-defined networks, Internet of Things, and application of artificial intelligence to cybersecurity. He won the Best Presentation Recognition at IEEE/ACM UCC 2013. He is a member of the IEEE since 2012 and Senior Member of the IEEE since 2015.

Ling Chen is a Research Engineer at Institute for Infocomm Research, A*STAR. She is a graduate student at National University of Singapore, School of Computing. She is interested in Cybersecurity, Homomorphic Encryptions and Deep Learning.



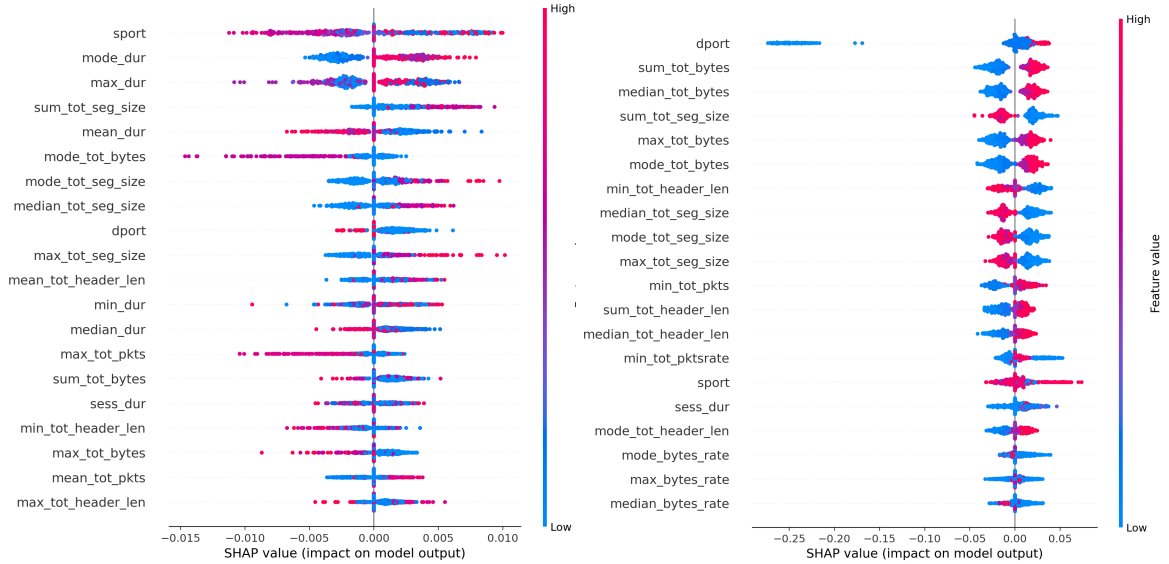
Luying Zhou received the B.S. and M.S. degree in Automatic Control in 1982 and 1985, respectively, from South China University of Technology, and the Ph.D. degree in Systems Engineering in 1990 from Xi'an Jiaotong University, China. From 1990 to 1995, he was a faculty member at South China University of Technology. He held a Postdoctoral position in SUNY at Buffalo and Syracuse University, New York from 1995 to 1998. Dr. Zhou has been a research scientist at the Institute for Infocomm Research, Singapore since 1998, and served as adjunct faculty member at NTU, Singapore from 2004 to 2012. He was a recipient of IEEE ICC 2012 best paper award. His research interests are in optical and wireless networks, and network security.

Sin G. Teo is a Scientist at Institute for Infocomm Research (I^2R), Agency for Science, Technology and Research (A*STAR), Singapore. He obtained the PhD degree from Monash University, Australia in 2016. His research interests include applied cryptography, data privacy and security, malware and network anomaly classification, federated learning, and deep learning.



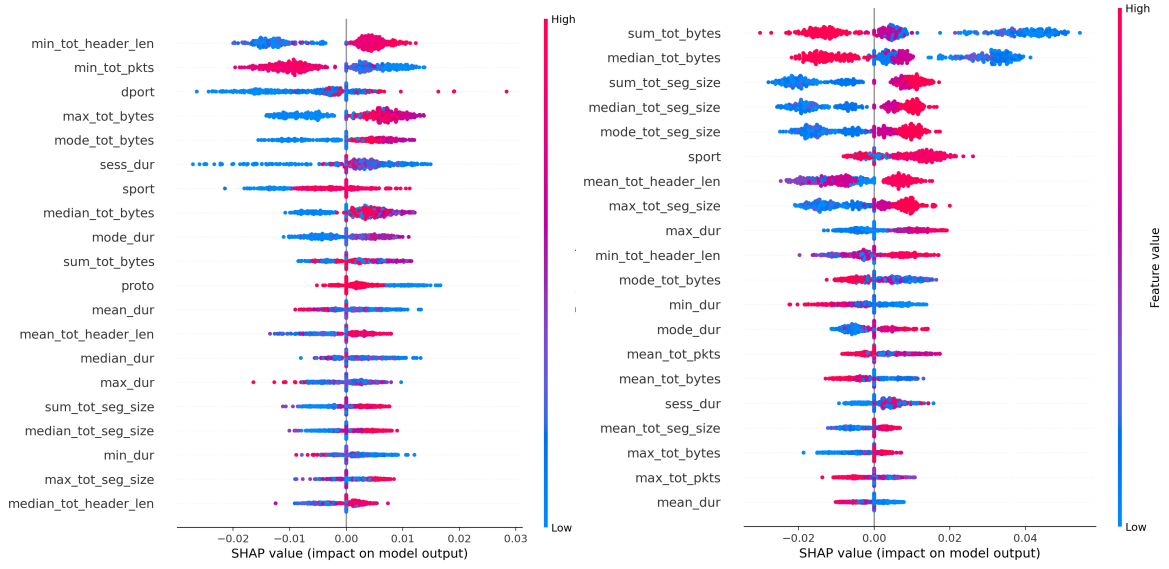
(a) Normal

(b) Andromeda



(c) Barys

(d) Emotet

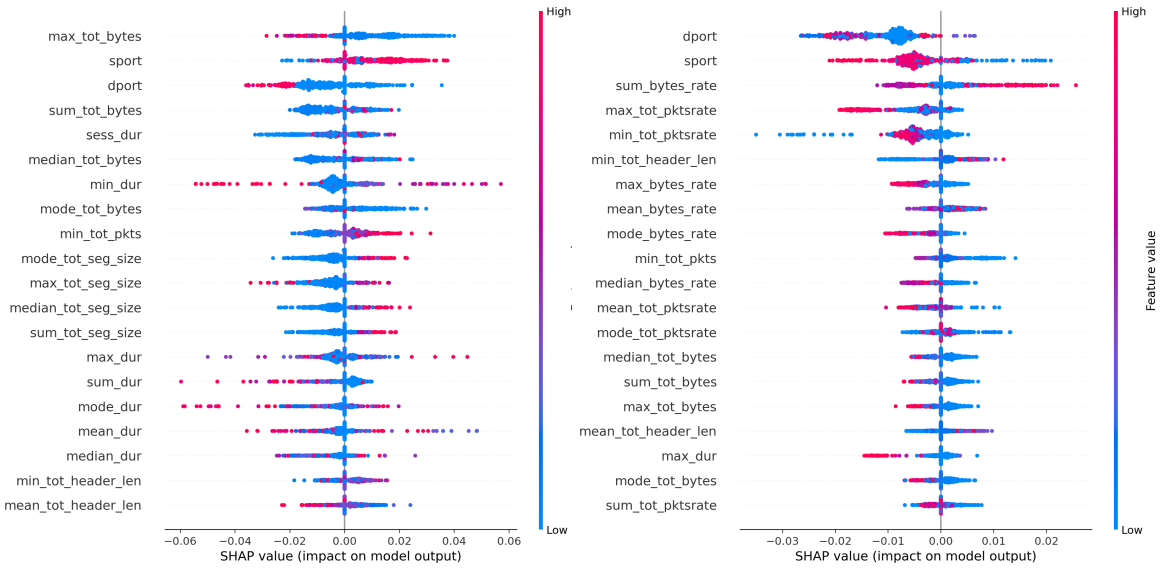


(e) Dridex

(f) Artemis

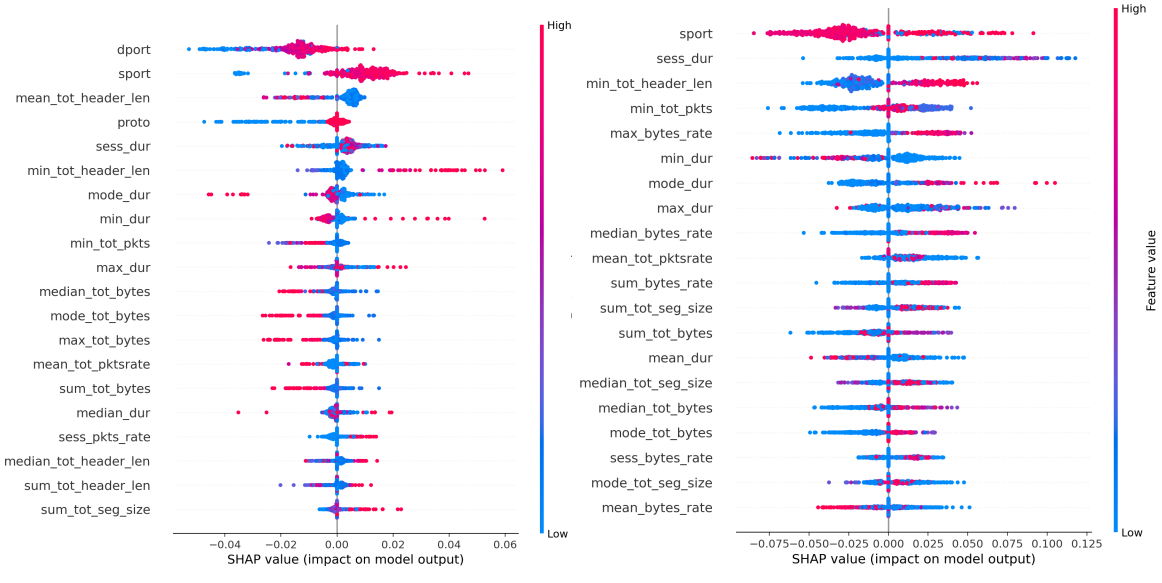
Fig. 12. The class-wise of top 20 features that have deep influence on the 1DCNN classification model output according to SHAP framework for DS_2. Due to space limitation we have plotted first 6 classes here.

This article has been accepted for publication in IEEE Transactions on Dependable and Secure Computing. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TDSC.2022.3183361



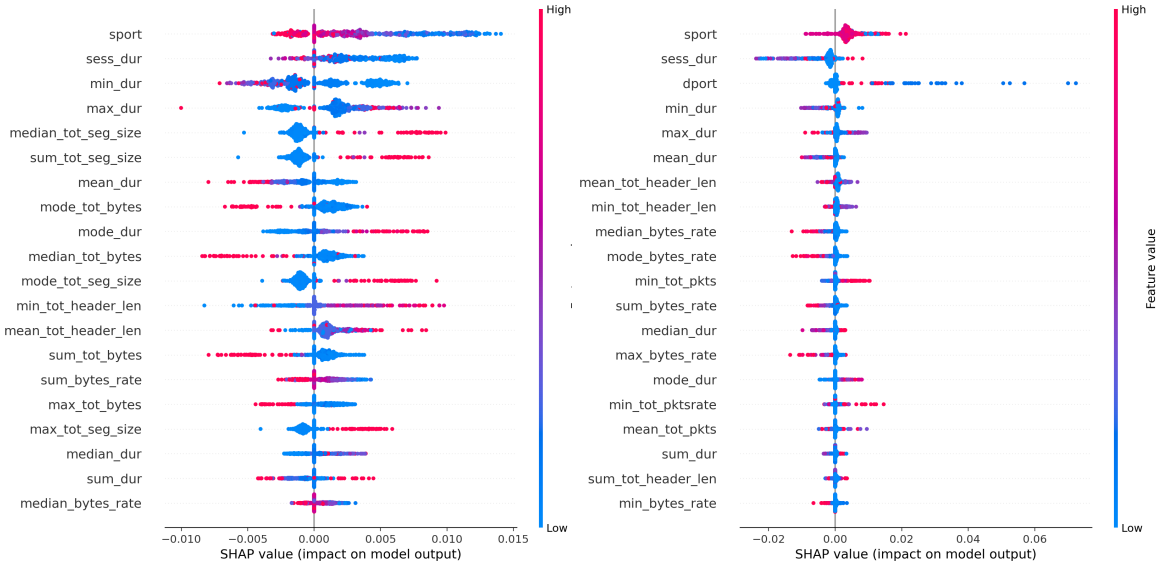
(a) Miuref

(b) Necurs



(c) Sality

(d) Trickbot



(e) Ursnif

(f) Wannacry

Fig. 13. The class-wise of top 20 features that have deep influence on the 1DCNN classification model output according to SHAP framework for DS_2. Due to space limitation we have plotted next 6 classes here.