IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 16 April 2024, accepted 9 June 2024, date of publication 1 July 2024, date of current version 9 July 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3421299

RESEARCH ARTICLE

Enhancing Explanation of LSTM-Based DDoS Attack Classification Using SHAP With Pattern Dependency

BASIL ASSADHAN[1], (Senior Member, IEEE), ABDULMUNEEM BASHAIWTH[1,2], AND HAMAD BINSALLEEH[3]

[1]Department of Electrical Engineering, King Saud University (KSU), Riyadh 11421, Saudi Arabia
[2]Department of Electronic Engineering and Communication, Hadhramout University (HU), Al Mukalla 50512, Yemen
[3]Department of Computer Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh 13318, Saudi Arabia

Corresponding author: Abdulmuneem Bashaiwth (abdulmuneem@student.ksu.edu.sa)

This work was supported by the International Scientific Partnership Program of King Saud University under Grant ISPP-18-134(2).

ABSTRACT DDoS attacks pose serious threats to the availability and reliability of computer networks. With the increasing complexity of DDoS attacks, the accurate detection and classification of these attacks is essential to ensure the protection of network systems. In this paper, we leverage the power of the LSTM model for DDoS attack classification and its ability to automatically learn complex patterns and select features from raw traffic at the packet level. LSTM models have remarkable performance in network traffic classification, however explaining the internal workings of them remains challenging, which hinders their wider adoption in real-world applications. To address this limitation, we propose the SHAP with Pattern Dependency (SHAPPD) approach to explain the predictions of the LSTM model. The results demonstrate significant performance in classifying the DDoS attacks from raw traffic using the LSTM model. SHAPPD effectively explains the predictions of the LSTM model, highlighting the underlying packet traffic fields that drive the LSTM to make its true and false positive predictions and finding the common fields between the DDoS attacks. The results of the comparison between the SHAPPD and the original SHAP emphasize that the SHAPPD is superior to the original SHAP in providing more elaborative justifications for DDoS attacks classification results. The SHAPPD, by quantifying the contribution of each input feature and considering the interdependencies between the features as well as the continued traffic packets, enables security analysts to gain insights into the decision-making process of the LSTM model and identify critical indicators about the DDoS attacks.

INDEX TERMS DDoS attacks, machine learning, DL classification, DL explanation, SHAP.

I. INTRODUCTION

In today's interconnected world, the threat of cyberattacks on network security is a significant and ever-growing concern to organizations, governments, and individuals. This leads to financial losses, data breaches, disruptions in critical services, and compromised privacy. One of the most common of today's cyberattacks is DDoS attacks. Distributed Denial of Service (DDoS) attacks pose significant threats to the availability and reliability of online services and networks [1]. These attacks exploit the inherent vulnerabilities of Internet's architecture and overwhelm targeted systems with a flood of malicious traffic, rendering them inaccessible to legitimate users [2]. DDoS attacks have become increasingly prevalent and sophisticated, necessitating robust defense mechanisms to mitigate their impact and ensure the uninterrupted operation of critical online infrastructures. DDoS attacks aim to disrupt the normal functioning of targeted systems by flooding them with a massive volume of traffic or exploiting vulnerabilities in network protocols [3]. The attackers typically harness a network of compromised computers, known as a botnet, to launch the attack [4]. This distributed nature makes it difficult to trace and block the attack traffic effectively. Additionally, modern DDoS attacks leverage various techniques, including amplification attacks, reflection attacks, and application-layer attacks, to overwhelm the

The associate editor coordinating the review of this manuscript and approving it for publication was Diego Oliva.

targeted systems and exhaust their resources [5]. The consequences of DDoS attacks can be severe including financial losses and service unavailability that leads to reputational damage for businesses and service providers [5]. Moreover, DDoS attacks can be used as a diversionary tactic to mask other malicious activities, such as data breaches or network intrusions, further complicating the security landscape [5].

To face the threats of DDoS attacks, researchers and practitioners have developed a range of defense strategies and mitigation techniques [6]. These techniques include network-level defenses, such as traffic filtering and rate limiting, as well as anomaly detection and traffic diversion mechanisms [6]. Furthermore, Machine Learning (ML) and Artificial Intelligence (AI) based approaches have been employed to detect and mitigate DDoS attacks by analyzing network traffic patterns and identifying anomalous behavior [7]. The transition from traditional machine learning to Deep Learning (DL) for detecting DDoS attacks has revolutionized the field by leveraging the power of neural networks to automatically learn complex patterns and features from network traffic data. Detecting DDoS attacks using DL models has gained significant attention due to the ability of these models to automatically learn complex patterns and features from raw network traffic data. DL techniques, such as Recurrent Neural Networks (RNNs), offer promising avenues for enhancing the accuracy and effectiveness of DDoS attack detection systems [8]. RNNs are well-suited for analyzing temporal dependencies in network traffic. By considering the sequential nature of packet arrivals, RNNs can capture long-term patterns and detect anomalies associated with DDoS attacks [9]. Long Short-Term Memory (LSTM) is a popular RNN architecture used for DDoS detection, as they can model both short-term and long-term dependencies in the traffic data [10].

DL models have achieved remarkable performance in complex tasks, but their black-box nature often raises concerns about transparency and trustworthiness. Interpreting DL models is crucial to gaining insights into their decision-making process, understanding the factors influencing their predictions, and ensuring their transparency and accountability. Therefore, researchers have developed explanation methods that aim to highlight how these models arrive at their predictions or decisions [10]. These explanation methods provide insights and explanations that help users understand the underlying factors that contribute to the model's outputs. Several explanation methods including [11], [12], [13], [14], [15], [16], have been proposed to interpret predictions of DL models. Some of these methods such as Local Interpretable Model-agnostic Explanations (LIME), LOcal Rule Explanation (LORD), Local Explanation Method using Nonlinear Approximation (LEMNA), and SHapley Additive exPlanation (SHAP) are used to explain DL-based security applications. These methods have limitations in providing robust and representative explanations of the RNNs that consider dependencies between the input samples as well as between the features within a single sample. The limitations include: 1) ignore the dependency among features of one input sample to the DL model and 2) ignore the dependency between the current input sample and the historical input samples. The classification problem for some applications such as image (each byte within an image independently represents the color of a pixel) is performed sufficiently by utilizing DL models that are based on a single input sample. The output of these models can be denoted by $y_t = f(\boldsymbol{x}_t)$, where $\boldsymbol{x}_t$ is a sample represented by a d-dimensional feature vector $(x_1, \ldots, x_d)^T$. On the other hand, DL such as RNN models that are based on the current input sample as well as k history inputs to make decisions are compatible with the security application that exhibit time-series inputs.The output of these models can be denoted by $y_t = f(x_t, x_{t-1}, \ldots, x_{t-k})$, that depends on the current input sample $\boldsymbol{x}_t$ and the k history inputs from x_{t-1} to x_{t-k}. As the network traffic is continuous traces through the time, they exhibit dependency among their packets as well as dependency among the features inside each packet. For instance, the fields in network packet headers have well-defined meanings dependencies; *TCP.flag* is a sub-feature of *TCP*, which means if the *TCP.flag* feature has impacts on the decision of DL model, then the *TCP* feature is impacted as well.

In our work, we propose an explanation approach SHAP with Pattern Dependency (SHAPPD) that develops the original SHAP method to improve its explanations by considering the dependencies among the network traffic packets as well as the interdependencies between the features within the packet. We apply the proposed approach to the LSTM model used to classify the DDoS attacks in the CICDDoS2019 dataset. The contribution of our work can be presented as follows:

1) Adopting the classification of the raw data instead of the featured data because of several benefits including: i) raw data classification retains the original information presented in the dataset without any specific transformations or feature engineering, ii) classifying raw data can help mitigating potential biases and overfitting issues that may arise from the selection and engineering of features, and iii) classifying raw data can enhance the explanation of the classification model's predictions where it becomes easier to understand how the model arrives at its decisions by analyzing the raw data.
2) Leveraging the capabilities of LSTM models to classify 12 classes of DDoS attacks, in addition to benign traffic directly from the raw data of the CICDDoS2019 dataset.
3) Proposing the SHAPPD approach that exploits the inherent continuity in the network traces to improve the LSTM model explanations by considering the interdependencies between network traffic traces, further enriching the explanations provided.

The remainder of the paper is organized as follows: Section II presents an overview of the related work in the domains of DDoS attack classification and DL models explanation. Section III outlines the preprocessing of the

CICDDoS2019 dataset [17]. Section IV describes our proposed approach (SHAPPD) in detail. Section V presents the results obtained from the experiments conducted on the CICDDoS2019 dataset classification and LSTM model explanation. The findings of the SHAPPD approach are analyzed and discussed in comparison with the original SHAP approach. Section VI provides a comprehensive conclusion, summarizing the main contributions of the study.

II. RELATED WORK

In this section, we review the prior literature on classification network traffic techniques, particularly those that use ML techniques, and explanation methods used to interpret DL models. We first present briefly several methods used to detect and classify network traffic and focus on the ones that employ ML to classify network traffic in its raw format. We then cover the works that implemented explanation methods to interpret DL models applied on raw network traffic.

A. NETWORK TRAFFIC CLASSIFICATION

There are several methods used to classify network traffic. We highlight the most popular of these methods including: port-based methods, payload inspection-based methods, and ML-based methods.

Port-based classification methods are a basic and well-known technique [18], which exploits information of the TCP/UDP packet's header to extract protocol port numbers. The extracted ports are compared with standardized ports of the Internet Assigned Numbers Authority (IANA) organization to perform classification purposes. The simplicity and the fastness in procedures of these methods qualified them to be used in firewalls and access control lists [18]. However, there are several factors that have a significant negative effect on the performance of this method. These factors include: port forwarding, pervasiveness of port obfuscation, protocol embedding, network address translation, and random port assignments.

Payload inspection-based methods, also known as Deep Packet Inspection (DPI), rely on the analysis of the application layer header and payload information [19]. These methods utilize what is known as predefined patterns, such as signatures for each involved protocol, [20], to distinguish the protocols from each other. The drawbacks of payload inspection methods include the violation of user privacy by accessing private information during payload analysis, and the predefined patterns used in inspection that needs to be updated continuously to capture the new abnormal traffic.

The aforementioned obstacles such as port forwarding, the pervasiveness of port obfuscation, violation of user privacy, and the continuous demand for updates limits the use of port and payload-based methods in modern network traffic classification Recent approaches to network traffic classification rely on ML techniques, which can deal with a wider range of network traffic [21]. However, the performance of ML-based approaches is highly based on the extracted features selected by humans which can limit the accuracy and generalizability. In addition, ML-based classification approaches usually need high storage and computational resources. Consequently, these demands restrict the utilization of ML-based classification approaches in resource-constrained fields [22]. As the network traffic classifier with real-time accuracy is the basis of Network Intrusion Detection systems (NIDs) and network management tasks, newer classification methods are needed. Therefore, the classification methods of network traffic using DL have emerged to avoid the difficult task of feature selection and gain feature information automatically during the training of the classifier [23]. One of the properties of classification approaches of network traffic using DL (e.g., Convolutional neural network (CNN) and RNN) is that they have a higher learning capability compared to the traditional ML methods (e.g., Random Forest (RF), Support Vector Machine(SVM), and K-Nearest Neighbors(KNN)) [21].

The process of feature extraction from network traffic requires preprocessing that involves utilizing various mathematical techniques to prepare network traffic for the DL model. These procedures of preprocessing can cause loss of information and affect the output of DL models. Moreover, as the features selected are often the outputs of other tools such as Intrusion Detection Systems (IDS), a bad selection of these features might enable adversarial perturbation of attack samples. This means that if the selected features do not capture the most relevant information or fail to capture the underlying patterns in the data, the model may rely heavily on less robust or easily fooled features. This can create opportunities for adversarial attacks, as the attacker can identify and manipulate those vulnerable features to deceive the model. To address these challenges, we directly apply the DL models to raw nibbles or bytes of a packet. In the following, we present recent works of DL-based network traffic classification from the raw data.

Hu and Shen [24], consider a DL method to classify intrusions in network traffic. Their proposed design utilizes the raw information of traffic as the features of flow and implements the hierarchical network structure of CNN and LSTM to automatically learn the spatial and temporal features of flow without involving feature engineering. The raw traffic packets are used to identify the intrusions where the design retains all the feature information of each traffic packet. They considered the first 10 packets from each flow and extracted only 160 bytes from each packet to represent the features. Therefore, there was 1,600-dimensional raw data for each flow. In this paper, CICIDS2017 dataset and CTU dataset were used to evaluate the proposed design. The results demonstrate that the proposed design could provide high detection classification performance (accuracy, precision, recall, and F1-score). In addition, the authors analyze the features that are significantly involved in intrusion traffic detection and give the true meanings of these important features. This was done by calculating the importance of the extracted features. Three different metrics were used: weight-based importance, gain-based importance, and cover-based importance. The final result was 11 features with top scores obtained by averaging the results of the three methods.

Zhang et al. [25], propose a framework to detect anomaly traffic and they call it D-PACK. This framework consists of using CNN and an autoencoder as an unsupervised deep learning model for auto-profiling detection classification. They claim that the D-PACK can detect the anomaly early by inspecting only the first few packets (80 bytes for each) in each network traffic flow. The authors use USTC-TFC2016 and Mirai-RGU datasets to evaluate the proposed framework. The results of experiments in this paper show that the D-PACK can detect the anomaly with high performance and low false positive rate even by exploiting the first packets for each flow.

Hwang et al. [26], develop an anomaly network traffic classification method called DataNet. This method is an encrypted data packet classifier that consists of a Multilayer Perceptron (MLP), stacked autoencoder (SAE), and CNN. The authors define a variable input of the DataNet classifier that is extended to 1,500 bytes per packet as the maximum value. To evaluate the DataNet, this paper used more than 20,000 data packets from 15 types of applications selected from an encrypted ISCX VPN-nonVPN dataset. The results illustrated that the developed DataNet can provide real-time detection and fine-grained awareness of harmful applications.

Wang et al. [27], propose an intrusion detection system called HAST-IDS. The hierarchical spatial-temporal features-based intrusion detection system HAST-IDS operates in a cascade manner using CNN and LSTM. HAST-IDS first exploits CNN to learn the low-level spatial network traffic features and then uses LSTM to learn high-level temporal features. These features are learned automatically using HAST-IDS without the need for feature engineering techniques and this contributes to reducing the false positive rate. The raw data (100-1,500 bytes per packet) from standard DARPA1998 and ISCX2012 datasets are used to evaluate HAST-IDS. The comparing results showed that the HAST-IDS exceeded the other approaches in classification performance particularly in terms of accuracy and false alarm rate.

Wang et al. [28], present a framework, Deep-Full-Range (DFR), to detect intrusions in encrypted network traffic. DFR uses three deep learning models; CNN, LSTM, and SAE, in the classification process. All these models are combined to fine classification of the encrypted traffic and provide a deep and full range understanding of the raw data input. Two public datasets, ISCX VPN-nonVPN traffic and ISCX 2012 IDS, are used to evaluate the DFR where 900 bytes for each packet have been adopted to represent the raw data input of the model. Authors, in [14], claim that their proposed framework (DFR) can outperform the state-of-the-art methods in terms of F1-score for both classification of encrypted traffic and intrusion classification.

Zeng et al. [29], propose a general framework for the classification of mobile and encrypted traffic using DL techniques. The proposed framework is based on a strict definition of its milestones such as the choice of the traffic object, the definition of the input, and the architecture of the DL model (CNN). The proposed framework is evaluated by three types of datasets: FB/FBM, Android, and iOS as raw inputs of 1D-CNN by performing two experiments. The first experiment uses a single model of 1D-CNN with the first 784 bytes of layer 4 payloads, while the second experiment uses multiple models of 1D-CNN with the first 576 bytes of layer 4 payloads. The authors consider their work as a starting point toward the design of effective mobile traffic classifiers. The result of this study was a DL-based traffic classification framework that was able to benefit from varied input data from mobile traffic and address multiple traffic classification tasks.

Aceto et al. [30], propose a network traffic classification approach, called Deep-Packet, using deep learning. Deep-Packet can integrate both feature extraction and classification phases in one scheme to deal with traffic characterization and application identifications. The architecture of Deep-Packet includes SAE and CNN that operate together for network traffic classification. The proposed framework is evaluated using raw data extracted from the UNB ISCX VPN-nonVPN dataset where 1,500 bytes of each IP packet were employed as input for the classification model. The results show that the Deep-Packet can achieve the best performance in terms of recall of 0.98 in the application identification task and 0.94 in the traffic categorization task.

Lotfollahi et al. [31], propose a new approach that combines a malicious classification using the LSTM model with a support word embedding technique. The proposed approach can extract packet semantic meanings and exploit the LSTM to learn the temporal relation among fields in the packet header and identify the behavior of inputs. The authors use ISCX2012, USTC-TFC2016, IoT dataset from Robert Gordon University, and IoT dataset collected on the Mirai Botnet to evaluate the proposed approach where the field in the packet header was considered as a word and trimmed to a fixed length of 54 bytes. The comparing results show that the proposed approach in this work can compete with the previous literature which classifies the malicious traffic at the flow level and this work can inspire the research community in terms of exploiting the advantages of DL to develop effective intrusion detection systems with significant detection rate.

The aforementioned studies about the classification of raw datasets using DL models have several limitations including: 1) Although the used dataset contains multiple attacks, the classification was limited only to binary classification, 2) some of the used datasets were out of date and have no recent attacks, 3) adopting the byte as the smallest unit to represent the raw dataset during the classification leads to missing some traffic packet information that may affect the classification decision, and 4) utilizing only the packet header information and ignoring the payload information might lead to less reliable classification. Therefore, in our work, we address these limitations through the multiclass classification of a recent dataset (CICDDoS2019) with a variety of DDoS attacks using the LSTM model which is suitable for classifying this type of dataset [10]. We adopt the nibble (4 bits) as the smallest unit of the raw data to ensure the utilization of all the traffic information. We also employ the packets' header and payload information in the classification process.

B. DL EXPLANATION METHODS

In this subsection, we present recent works that used explanation methods to interpret the predictions of DL models, in particular, the SHAP method as it is the focus of this paper.

Hwang et al [32], develop an approach to address several issues that make the classification of DDoS attacks in CICDDoS2019 dataset using ML models less efficient. These issues include the existence of irrelevant dataset features, class imbalance, and lack of transparency of the classification model. They first preprocessed CICDDoS2019 and use the adaptive synthetic oversampling technique to address the imbalance issue. They then conduct a selection mechanism for the dataset features through embedding SHAP importance to eliminate recursive features with Decision Tree (DT), Random Forest (RF), Gradient Boosting (GBoost),Light Gradient Boosting (LGBoost), and Extreme Gradient Boosting (XGBoost) models.

After that, LIME and SHAP explanation methods are performed on the dataset with selected features to ensure model transparency. Finally, binary classification is performed by feeding the selected features to K-Nearest Oracle Eliminate (KNORA-E) and K-Nearest Oracle Union (KNORA-U) dynamic ensemble selection techniques. The classification experiment is performed on balanced and imbalanced datasets. The findings show that the balanced dataset performance outperformed the imbalanced datasets. The authors stated that using KNORA-E and KNORA-U improved the classification performance in terms of accuracy to 99.9878% with KNORA-E and 99.9886% with KNORA-U compared to using the classification approach without KNORA.

Batchu and Seetha [33], propose ensemble tree models approach, DR and RF, to improve IoT-IDSs performance that evaluated on three IoT-based IDS datasets (IoTID20, NF-BoT-IoT-v2, and NF-ToN-IoT-v2). They assert that their proposed approaches provide 100% performance in terms of accuracy and F1 score compared to other methods of the same used datasets while they demonstrate lower Area Under the ROC Curve (AUC) compared to previous Deep FeedForward(DFF) and RF methods using the NF-ToN-IoT-v2 dataset. The authors also exploit the SHAP method in both global and local explanations. The global explanation was used to interpret the model's general characteristics by analyzing all its predictions by the heatmap plot technique. On the other hand, the local explanation was used to interpret the prediction results of each input (instance) of the model using the decision plot technique.

Le et al., [34], propose the SPIP (S: Shapley Additive exPlanations, P: Permutation Feature Importance, I: Individual Conditional Expectation, P: Partial Dependence Plot) framework to assess explainable DL models for IDS in IoT domains. They implement LSTM model to conduct binary and multiclassification in three datasets: NSL-KDD, UNSW-NB15 and ToN-IoT. The predictions of LSTM model were interpreted locally and globally using SHAP, PFI, ICE, and PDP explanation methods. The proposed approach is able to extract a customized set of input features that can outperform the original set of features in the three datasets and enhance the utilization of AI-based IDS in cybersecurity systems. The results show that the explanations of the proposed method depend on the performance of IDS models. This indicates that the framework's performance is affected negatively in the presence of poorly built IDS which causes the proposed framework to miss detecting the exploited vulnerability.

Keshk et al., [35], introduce a standard to compare and assess the explanation methods. They classified the investigated explanation methods into black-box methods and white box methods. Six explanation methods (LIME, SHAP, LEMNA, Gradients, IG, and LRP) were investigated and evaluated. The evaluation metrics: completeness, stability, efficiency, and robustness were implemented in this work. The authors applied the DL model RNN to four selected security systems (Drebin+, Mimicus+, DAMD, and VulDeePecker) to provide a diverse view of security. They construct general recommendations to select and utilize explanation methods in network security from their observations of significant differences between the methods.

Warnecke et al., [36], propose principled instructions to evaluate the quality of the explanation methods. Five explanation approaches (LIME, Anchor, LORE, SHAP, and LEMNA) were investigated. These approaches were applied to detect Android malware and identify their family. The authors design three quantitative metrics to estimate stability, effectiveness, and robustness. These metrics are principal properties that an explanation approach should fulfill for crucial security tasks. The results show that the evaluation metrics can evaluate different explanation strategies and enable users to learn about malicious behaviors for accurate analysis of malware.

The aforementioned studies employ explanation techniques like LIME, SHAP, and LEMNA to interpret predictions made by DL models. However, these explanation methods have certain limitations when it comes to providing comprehensive and accurate explanations for DL models, especially recurrent neural network (RNN) models, which take into account dependencies among input samples and features within a single sample. These limitations include disregarding the interdependence among features within an individual input sample and the relationship between the current input sample and the historical input samples. Consequently, in our research, we propose the SHAPPD approach, which addresses these limitations of the SHAP method by considering both the dependency between the features of an input sample and the correlation between multiple input samples. This approach aims to enhance the quality of the resulting explanations compared to that resulting from the original SHAP.

III. PREPROCESSING OF CICDDOS2019 DATASET

Understanding the intricate details of DDoS attacks is essential to develop effective mitigation and defense mechanisms against such malicious activities. Such details include the techniques used in the attacks and the impact of the attacks on network resources.

The CICDDoS2019 dataset [17] addresses this need by providing a diverse set of DDoS attack instances, encompassing a wide range of attack vectors and strategies. The CICDDoS2019 dataset was developed by the Canadian Institute for Cybersecurity at the University of New Brunswick [17]. This dataset is a comprehensive and valuable resource to aid researchers, practitioners, and professionals in the cybersecurity field to understand and mitigate DDoS attacks. The CICDDoS2019 dataset provides a rich collection of real-world DDoS attack scenarios, capturing various attack types, network traffic patterns, and attack characteristics.

CICDDoS2019 is classified into two main categories, Reflection-based DDoS and Exploitation-based DDoS. The attacks in reflection-based DDoS are subcategorized into TCP (MSSQL, SSDP), UDP (NTP, TFTP), or TCP/UDP (DNS, LDAP, SNMP, WebDDoS). While exploitation-based DDoS, in concept, is similar to Reflection-based DDoS with the difference that these attacks can be conducted through application-layer protocols using transport-layer protocols. The attacks evidenced in this category are subcategorized into TCP (SYN-Flood) or UDP (UDP Flood, UDP-Lag). The CICDDoS2019 dataset is available in two formats; the raw data in format PCAP files, and the extracted features flows in format CSV files. As in our work we target the classification of raw dataset, we use the PCAP files of CICDDoS2019. There are 819 separated PCAP files each with about 195 KB to cover about 12 classes of DDoS attacks.

We first used 'tshark' tool to convert these files into corresponding JSON files that contain raw information of each network packet in hexadecimal format. We then built the corresponding CSV files. As the inputs of the DL model should be constant in length, each row (sample) in the CSV file is a vector of packet raw that consists of 324 elements. Each element represents a decimal number (0 - 15) corresponding to sequential 4 bits (nibble). To maintain each sample with a constant length (324), we trim the long packets and pad the short ones. The sample with a length of 324 can cover the header packet and a portion of the payload data. It is noteworthy to state that we remove Ethernet information from each packet, which consists of 14 bytes (28 nibbles) represented by the red rectangular in Figure 1. Therefore, to get 324 nibbles, we start by nibble number (28) and end by nibble number (351).

We finally labeled each sample of CSV files with an attack name according to the execution time of the attack as shown in Table 1. We used the packet timestamp to figure out the start and end time for each attack. We note that the # of samples column of Table 1 shows how the number of samples varies significantly between different attacks, where its maximum value is 75,517,782 for the NTP class and its

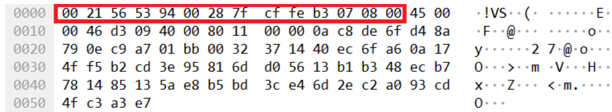

FIGURE 1. Screenshot from the raw data displayed using Wireshark. The part inside the red rectangular represents the Ethernet information, which is removed from the used data.

TABLE 1. Number of samples for each DDoS attack in the CICDDoS2019 dataset that was obtained during the execution time for each attack.

Attack	Execution time	# of Samples
PortMap	09:43 - 09:51	3,920
NTP	10:35 - 10:45	75,517,782
DNS	10:52 - 11:05	1,583,145
LDAP	11:22 - 11:32	9,250,592
MSSQL	11:36 - 11:45	7,102,655
SNMP	12:12 - 12:23	16,882,478
SSDP	12:27 - 12:37	9,446,212
UDP	12:45 - 13:09	10,903,442
UDP-Lag	13:11 - 13:15	20,849
Web	13:18 - 13:29	7,339
SYN	13:29 - 13:34	2,576,862
TFTP	13:35 - 17:15	68,562,992

TABLE 2. Number of samples in each class of the CICDDoS2019 dataset after the data balancing process.

Attack	# Samples	Attack	# Samples
PortMap	3,920	NTP	20,849
DNS	20,849	LDAP	20,849
MSSQL	20,849	SNMP	20,849
SSDP	20,849	UDP	20,849
UDP-Lag	20,849	Web	7,339
SYN	20,849	TFTP	20,849

FIGURE 2. An example of one sample of DL model input that ranges from element F0 to F323 along with their decimal values and is followed by the attack label.

minimum is 3,920 for the PortMap class. Training the classifier with such imbalanced data is not practical and results in low classification performance. Therefore, we balance the data by taking samples from the classes with large number of samples (DNS, MSSQL, SSDP, SYN, NTP, LDAP, SNMP, UDP, and TFTP) to be balanced with UDP-Lag (20,849 samples) and leaving the low classes (PortMap and Web) without changes. As the attack and benign samples were generated from different machines [17], the attack samples have different IP source. We utilize the timestamp and IP source for each sample to ensure that the sample selection is inside the intended execution time. Table 2 presents the balanced CICDDoS2019 classes.

The final version of each sample in a CSV file is a feature-vector ranging from F0 to F323 and followed by the attack label. Figure 2 shows an example of a sample form NTP CSV file. We concatenated the CSV files of classes to one file that contains 240,598 samples covering 12 DDoS attacks plus benign traffic. The dataset matrix (N, m) as a CSV file is shown in the left side of Figure 3 where N represents the number of samples, while m represents the number of features (F0 - F323). Each sample in the dataset is associated with the class label y. To consider the continuity in the dataset, we adopt the timesteps window T samples with $(T - 1)$ as a sliding window during the whole dataset. This procedure converts our 2D matrix (N, m) into 3D matrix $((N - T) + 1, T, m)$ while maintaining the corresponding label for the last sample of T timesteps as shown in Figure 3.

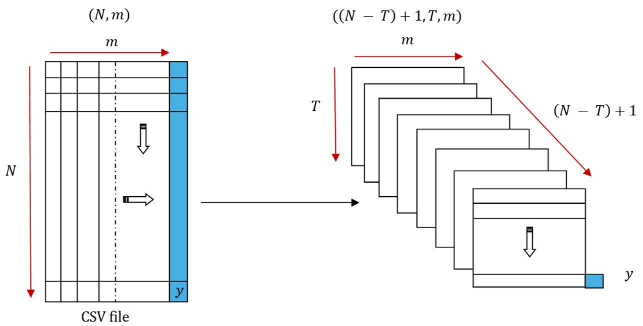

FIGURE 3. Implementation of the packet continuity in the CICDDoS2019 dataset by adopting timesteps window T samples with $(T - 1)$ sliding window.

IV. PROPOSED APPROACH

RNN models such as LSTM are based on the current input sample as well as several history inputs to make their decisions. These models are compatible with the security application that exhibits time-series inputs such as network traffic. As the network traffic is continuous traces through time, they manifest dependency among their packets as well as dependency among the features inside each packet. Therefore, the explanation method of the LSTM models that adopts the interactions between the features and the packet traces is more robust and representative.

SHapley Additive exPlanation (SHAP) is one of the explanation methods that can explain the predictions of the black box models such as LSTM. The SHAP method was proposed by Lundberg and Lee [14] to explain the black box models based on calculating the Shapley Values [38]. Shapley value represents the contribution of each feature of the input instance in predicting the black box model.

SHAP method considers only the dependency among the packet features and ignores the dependency between the input samples where the Shapley values are calculated by determining the marginal contribution of each feature. The marginal contribution measures how the prediction changes when a feature is added or removed from the combination. The drawback of the SHAP method, particularly in the time-series application, is ignoring the dependency between the current explained input sample and the history inputs. This drawback leads the SHAP method to provide an explanation that represents only the effect of the features in individual samples on the model prediction ignoring the interaction between the intended sample and the history samples.

In this paper, we propose the SHAP with Pattern Dependency (SHAPPD) approach that can develop the SHAP method to consider explanations of the current input sample as well as the $(T - 1)$ history input samples. Figure 4 shows an architecture of the LSTM explanation using the SHAPPD approach including three parts: test set instances, LSTM model, and SHAPPD model. Algorithm 1 illustrates how these three parts works to produce the explanations.

Algorithm 1 Procedures of SHAPPD

Input: $\boldsymbol{I} = \{I_1, I_2, I_3, \ldots\ldots, I_n\}$ are n class instances from test set and I_i is $(T \times m)$ array: m is number of features in the dataset and T is number of continuous packets.

Output: S is a set of important features that represents the explanation.

$ALL_F = [\,]$,

for $I_i \in \boldsymbol{I}$ **do**

$V_{T\times m} = Expl(I_i)$, where $V_{T\times m}$ are ± Shapley values $(T \times m)$ of input features.

$C_{Expl} = \{Fl, \ldots\}\forall Fl{:}Fl$ has +max(Shapley value) and l $\in \{0, 1, 2, \ldots., m\}$.

$ALL_F \longleftarrow C_{Expl}$

end for

return $S = \{Fl, \ldots\}\forall Fl$: Fl occurs in each C_{Expl} of ALL_F

The input $\boldsymbol{I}$ of Algorithm 1 is the number of instances from the testset with TP or FP prediction of the certain class where each instance is a $(T \times m)$ array. On the other hand, the output of the SHAPPD algorithm S is a set of important features that represents the explanation of the prediction of the LSTM model on the input instances. Instead of explaining the samples one by one as in the standard SHAP, our approach (SHAPPD) can explain T samples simultaneously to exploit the dependency between the continues samples (packets) in the series data and produce representative explanation. The output V_{Tm} of explaining each instance in $\boldsymbol{I}$ is positive or negative $(T \times m)$ Shapley values of the m features as shown in Figure 4, Step 1. Then, only the features with positive and maximum Shapley value are selected, as shown in Figure 4, Step 2, as they push the baseline value of the SHAP toward the interested prediction. This step in Algorithm 1 produces the vector C_{Expl} that contains all features with +max (Shapley values). Since the input $\boldsymbol{I}$ has n instances, we obtain n of C_{Expl} vectors as shown in Figure 4, Step 3. All these vectors are added to the ALL_F list. The last step of our Algorithm provides a set of features in a vector S that represent the explanation of the intended class. This vector is extracted from the n explanations in the previous step by selecting the repeated features as shown in Figure 4, Step 4. The repeated features mean that each feature in S occurs in all the n explanations, which indicates that the selected features in S are more robust and representative of the class explanation.

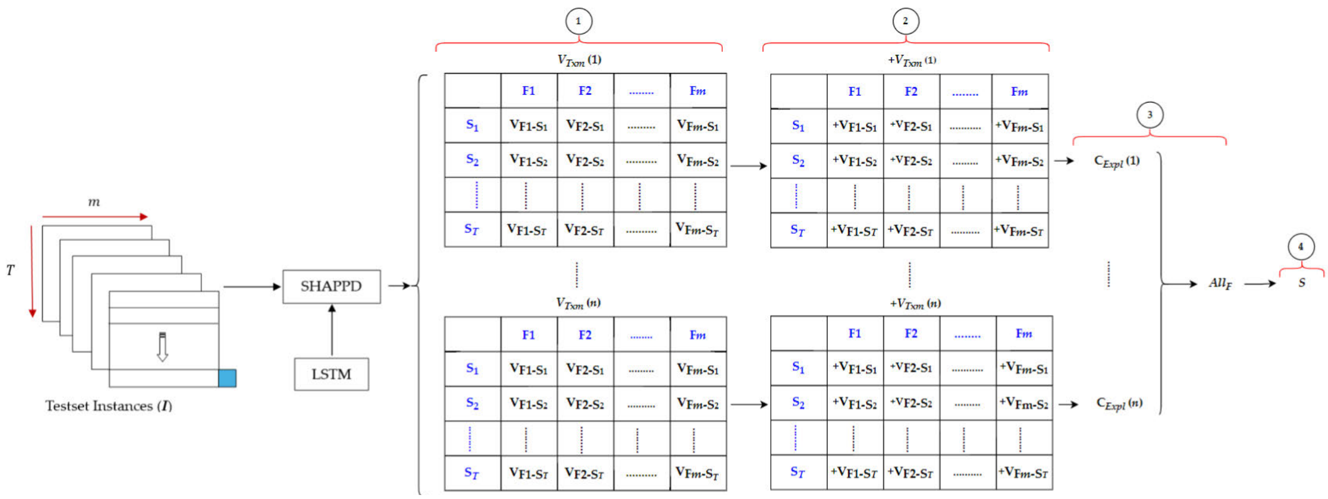

FIGURE 4. The architecture of the LSTM explanations using the SHAPPD approach where the predictions of T samples each with m nibbles are explained simultaneously to produce an explanation vector S of n inputs each with T samples.

V. RESULT AND DISCUSSION

In this section, we first set the architecture of experiments that are performed in this paper. We then conduct two experiments including DDoS attacks classification using the LSTM model and explain this model using the SHAPPD and the original SHAP. The results from these experiments are presented and discussed. We also compare the result of the SHAPPD to the original SHAP for evaluation purposes. We finally justify the LSTM TP predictions depending on the resulting explanations.

A. EXPERIMENTAL SETUP

Figure 5 shows the architecture used to perform the classification and explanation experiments. This architecture consists of three parts: dataset preprocessing, classification model, and explanation model.

The dataset preprocessing part was explained in detail in Section III. The output of the dataset preprocessing part is divided into train and test sets at a ratio of 70% and 30% respectively. The LSTM model uses the train set to learn the classification patterns automatically from the raw data and evaluates its predictions using the test set.

In this paper, the design of the LSTM model consists of two LSTM layers (`LSTM_1` and `LSTM_2`) and two Dense layers (`Dense_1` and `Dense_2`) as shown in Figure 6. The LSTM layers and the first Dense layer include 64 hidden neurons for each, while the second Dense layer includes 13 neurons corresponding to the model output. The neuron activation function of each layer uses the `RELU` function for nonlinear operation except the second Dense uses the `Softmax` function to compute the probability for each class during the classification process. We use two dropout layers with a rate of 0.2 and one batch normalization layer to prevent overfitting, improve the generalization capabilities of the model, and help to stabilize and accelerate the training process. The learning rate of the `Adam` optimizer in the designed model is 0.001 and the loss function is `categorical cross-entropy`. The model learns the temporal features of all samples inside the time window T and exploits the time dependency between them to enhance its predictions. To show the impact of using T on the model performance, we perform several experiments to train the model by increasing the values of T incrementally from 1 to 10. The results of the experiments demonstrate that the increase in the value of T leads to improved model performance. We settled for 10 samples to avoid the complexity of our proposed approach specifically in the explanation part.

TABLE 3. Overall metrics used to evaluate the performance of the LSTM classification model.

Metric	Expression	Metric	Expression
ACC	$\frac{1}{T}\sum_{i=1}^{C} TP_i$	PR	$\frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i+FP_i}$
RE	$\frac{1}{C}\sum_{i=1}^{C}\frac{TP_i}{TP_i+FN_i}$	F1-score	$\frac{1}{C}\sum_{i=1}^{C}\frac{2\times PR_i\times RE_i}{PR_i+RE_i}$
FPR	$\frac{1}{C}\sum_{i=1}^{C}\frac{FP_i}{FP_i+TN_i}$	FNR	$\frac{1}{C}\sum_{i=1}^{C}\frac{FN_i}{TP_i+FN_i}$

To evaluate the performance of the LSTM model we use the general performance metrics including Accuracy (ACC), Precision (PR), Recall (RE), F1-score, False Positive Rate (FPR), and False Negative Rate (FNR). We provide the following definitions to build the performance metrics:

- TP_i: Number of instances correctly classified with the label of i,
- TN_i: Number of instances correctly classified with the labels not i,
- FN_i: Instances classified as label i, but they belong to another class, and
- FP_i: Instances of i, but mistakenly classified in another class.

Table 3 shows the formal expression of overall performance metrics: ACC, PR, RE, F1-score, FPR, and FNR that are used to evaluate the LSTM model. N_c is the number of all samples in the testset and C is the number of classes in the dataset.

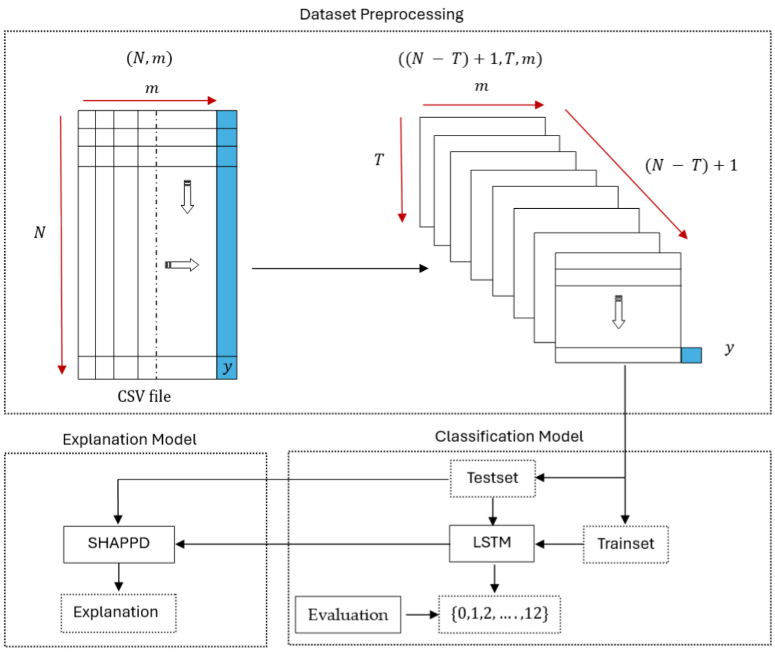

FIGURE 5. The overall architecture used in experimental results including dataset preprocessing, DDoS attack classification using the LSTM, and explanation of the LSTM using the SHAPPD.

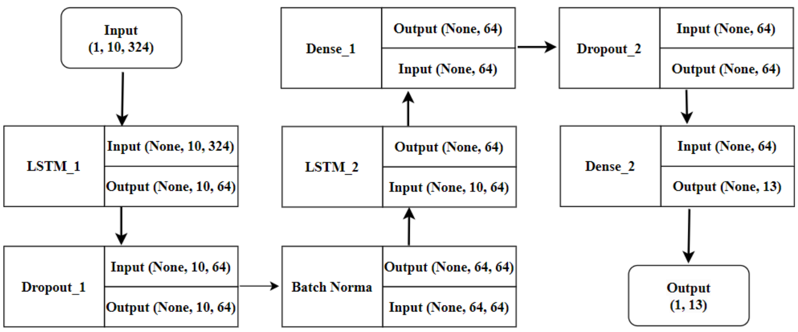

FIGURE 6. The design of the LSTM model shows the estimated layers: the input layer, the output layer, and the necessary hidden layers.

TABLE 4. The performance metrics of multiclass classification on the CICDDoS2019 dataset using the LSTM model.

Class	ACC	PR	RE	F1-score	FPR	FNR
Benign	0.99	0.95	1.00	0.97	0.0041	0.0053
DNS	0.99	0.98	1.00	0.99	0.0013	0.0010
LDAP	1.00	1.00	1.00	1.00	0.0001	0.0006
MSSQL	1.00	1.00	1.00	1.00	0.0000	0.0002
NTP	0.99	1.00	0.93	0.97	0.0000	0.0607
PortMap	0.99	0.99	0.94	0.96	0.0010	0.0289
SNMP	1.00	1.00	1.00	1.00	0.0003	0.0023
SSDP	1.00	1.00	1.00	1.00	0.0003	0.0024
SYN	1.00	1.00	1.00	1.00	0.0000	0.0020
TFTP	1.00	1.00	1.00	1.00	0.0000	0.0002
UDP	1.00	0.99	1.00	0.99	0.0006	0.0033
UDP-Lag	0.99	0.99	1.00	0.99	0.0009	0.0137
Web	0.99	0.95	0.94	0.99	0.0040	0.0281

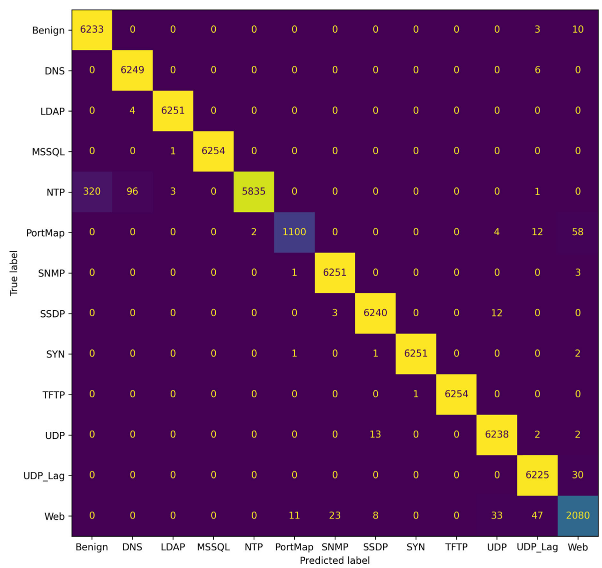

FIGURE 7. The Confusion matrix resulting from the multiclass classification of the CICDDoS2019 dataset using the LSTM model.

The third part of Figure 5 is the explanation of the LSTM model using our proposed approach (SHAPPD) explained in Section IV. To evaluate the SHAPPD approach, we compare

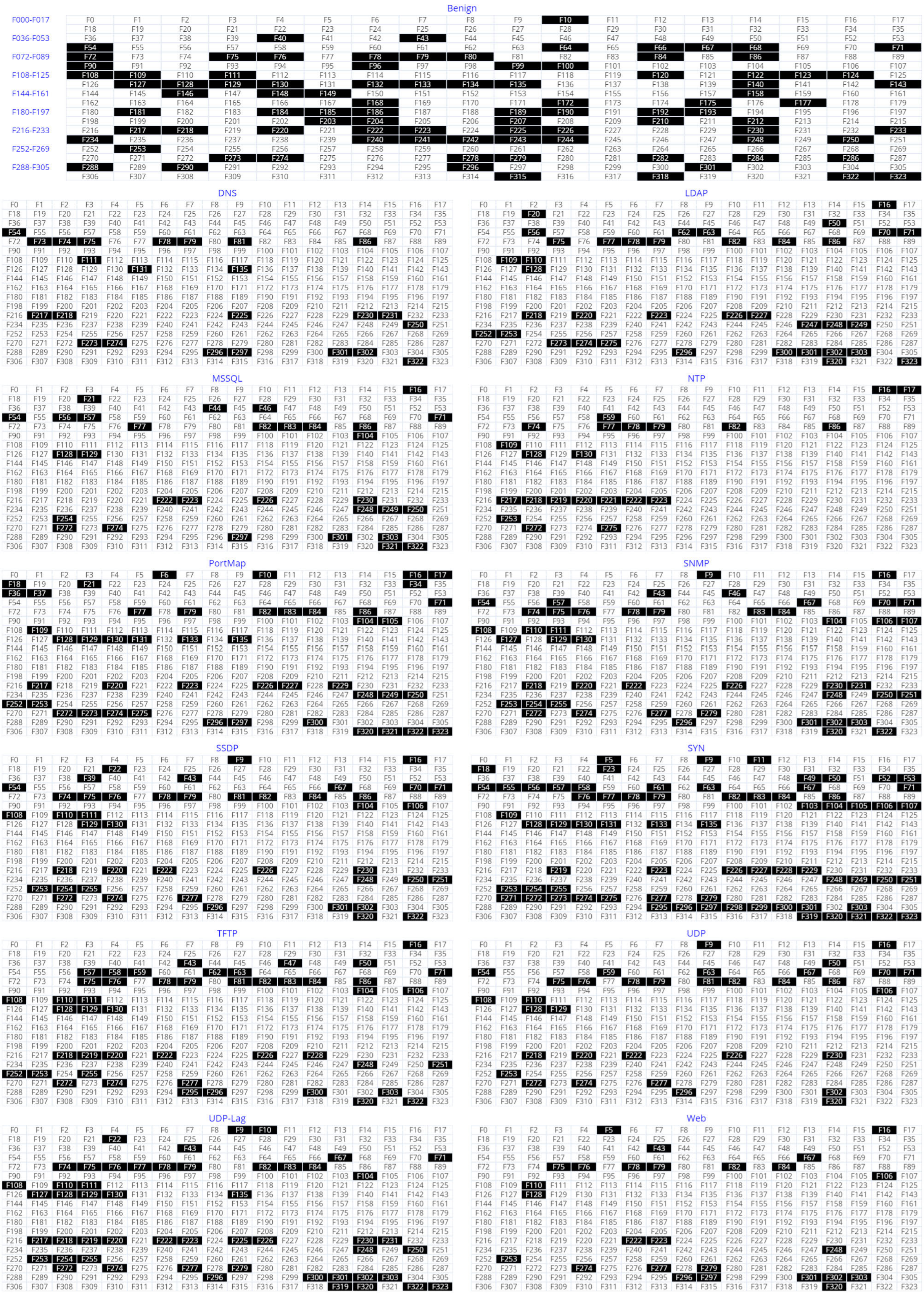

FIGURE 8. Explanations of TP samples of all classes in the CICDDoS2019 dataset where the shaded cells with black represent the features\nibbles that interpret the LSTM predictions according to the SHAPPD approach.

TABLE 5. SHAPPD explanations of TP samples for each class in the CICDDoS2019 dataset. The features\nibbles for each class are the shaded cells with black in Figure 8.

Class (#)	Features\Nibbles Name
Benign (92)	’F99’, ’F79’, ’F207’, ’F96’, ’F100’, ’F75’, ’F54’, ’F279’, ’F296’, ’F66’, ’F76’, ’F274’, ’F230’, ’F322’, ’F68’, ’F40’, ’F323’, ’F301’, ’F225’, ’F10’, ’F318’, ’F233’, ’F64’, ’F241’, ’F172’, ’F71’, ’F143’, ’F248’, ’F218’, ’F127’, ’F186’, ’F140’, ’F148’, ’F67’, ’F111’, ’F217’, ’F158’, ’F120’, ’F203’, ’F189’, ’F286’, ’F43’, ’F315’, ’F168’, ’F90’, ’F240’, ’F192’, ’F80’, ’F108’, ’F177’, ’F226’, ’F135’, ’F193’, ’F243’, ’F181’, ’F72’, ’F288’, ’F133’, ’F149’, ’F124’, ’F175’, ’F210’, ’F84’, ’F185’, ’F204’, ’F278’, ’F132’, ’F122’, ’F222’, ’F220’, ’F212’, ’F242’, ’F282’, ’F128’, ’F244’, ’F290’, ’F123’, ’F234’, ’F190’, ’F184’, ’F284’, ’F130’, ’F78’, ’F250’, ’F129’, ’F109’, ’F273’, ’F223’, ’F253’, ’F86’, ’F134’, ’F146’
DNS (24)	’F131’, ’F302’, ’F250’, ’F217’, ’F75’, ’F225’, ’F81’, ’F111’, ’F297’, ’F231’, ’F301’, ’F296’, ’F79’, ’F273’, ’F73’, ’F54’, ’F274’, ’F74’, ’F78’, ’F218’, ’F230’, ’F135’, ’F86’, ’F322’
LDAP (38)	’F56’, ’F63’, ’F86’, ’F79’, ’F50’, ’F82’, ’F16’, ’F128’, ’F62’, ’F275’, ’F78’, ’F109’, ’F77’, ’F70’, ’F71’, ’F223’, ’F249’, ’F252’, ’F296’, ’F248’, ’F220’, ’F323’, ’F274’, ’F320’, ’F110’, ’F253’, ’F301’, ’F226’, ’F75’, ’F84’, ’F20’, ’F218’, ’F247’, ’F300’, ’F227’, ’F302’, ’F303’, ’F273’
MSSQL (31)	’F56’, ’F77’, ’F249’, ’F21’, ’F297’, ’F16’, ’F272’, ’F86’, ’F321’, ’F71’, ’F128’, ’F248’, ’F223’, ’F222’, ’F129’, ’F46’, ’F82’, ’F301’, ’F44’, ’F104’, ’F54’, ’F274’, ’F84’, ’F83’, ’F303’, ’F254’, ’F226’, ’F322’, ’F57’, ’F250’, ’F230’
NTP (22)	’F79’, ’F74’, ’F78’, ’F217’, ’F220’, ’F222’, ’F218’, ’F82’, ’F59’, ’F16’, ’F272’, ’F77’, ’F128’, ’F219’, ’F17’, ’F109’, ’F221’, ’F86’, ’F275’, ’F223’, ’F253’, ’F130’
PortMap (47)	F129’, ’F130’, ’F77’, ’F36’, ’F16’, ’F34’, ’F37’, ’F109’, ’F320’, ’F105’, ’F131’, ’F135’, ’F322’, ’F21’, ’F227’, ’F79’, ’F321’, ’F6’, ’F10’, ’F86’, ’F17’, ’F18’, ’F71’, ’F104’, ’F220’, ’F82’, ’F223’, ’F253’, ’F249’, ’F248’, ’F296’, ’F250’, ’F133’, ’F252’, ’F128’, ’F297’, ’F272’, ’F275’, ’F226’, ’F323’, ’F229’, ’F300’, ’F217’, ’F84’, ’F274’, ’F83’, ’F273’
SNMP (47)	’F302’, ’F16’, ’F67’, ’F78’, ’F279’, ’F57’, ’F71’, ’F76’, ’F320’, ’F79’, ’F106’, ’F296’, ’F43’, ’F129’, ’F75’, ’F272’, ’F108’, ’F274’, ’F231’, ’F127’, ’F220’, ’F222’, ’F277’, ’F110’, ’F74’, ’F251’, ’F130’, ’F230’, ’F9’, ’F250’, ’F322’, ’F303’, ’F104’, ’F70’, ’F301’, ’F218’, ’F248’, ’F84’, ’F226’, ’F255’, ’F83’, ’F111’, ’F253’, ’F54’, ’F107’, ’F46’, ’F254’
SSDP (44)	’F39’, ’F79’, ’F70’, ’F67’, ’F75’, ’F74’, ’F71’, ’F78’, ’F296’, ’F43’, ’F277’, ’F302’, ’F16’, ’F76’, ’F274’, ’F320’, ’F220’, ’F110’, ’F255’, ’F230’, ’F86’, ’F9’, ’F81’, ’F253’, ’F218’, ’F129’, ’F301’, ’F22’, ’F250’, ’F111’, ’F272’, ’F248’, ’F104’, ’F222’, ’F84’, ’F108’, ’F82’, ’F106’, ’F251’, ’F226’, ’F54’, ’F322’, ’F130’, ’F254’
SYN (71)	’F79’, ’F67’, ’F103’, ’F272’, ’F78’, ’F83’, ’F82’, ’F279’, ’F76’, ’F57’, ’F274’, ’F86’, ’F5’, ’F296’, ’F84’, ’F129’, ’F128’, ’F18’, ’F301’, ’F297’, ’F277’, ’F226’, ’F320’, ’F322’, ’F49’, ’F54’, ’F9’, ’F223’, ’F323’, ’F250’, ’F105’, ’F104’, ’F135’, ’F248’, ’F227’, ’F71’, ’F319’, ’F249’, ’F56’, ’F254’, ’F251’, ’F107’, ’F61’, ’F53’, ’F255’, ’F106’, ’F11’, ’F133’, ’F273’, ’F321’, ’F253’, ’F58’, ’F303’, ’F130’, ’F63’, ’F229’, ’F299’, ’F131’, ’F23’, ’F50’, ’F295’, ’F271’, ’F77’, ’F55’, ’F52’, ’F228’, ’F275’, ’F219’, ’F298’, ’F300’, ’F109’
TFTP (47)	’F78’, ’F220’, ’F79’, ’F218’, ’F76’, ’F58’, ’F296’, ’F75’, ’F57’, ’F82’, ’F81’, ’F59’, ’F277’, ’F86’, ’F228’, ’F110’, ’F106’, ’F320’, ’F222’, ’F111’, ’F43’, ’F251’, ’F130’, ’F129’, ’F16’, ’F62’, ’F272’, ’F84’, ’F71’, ’F253’, ’F274’, ’F63’, ’F226’, ’F104’, ’F50’, ’F255’, ’F108’, ’F128’, ’F300’, ’F303’, ’F83’, ’F248’, ’F295’, ’F219’, ’F47’, ’F252’, ’F322’
UDP (34)	’F79’, ’F16’, ’F81’, ’F272’, ’F78’, ’F71’, ’F253’, ’F274’, ’F63’, ’F82’, ’F302’, ’F76’, ’F110’, ’F106’, ’F128’, ’F129’, ’F230’, ’F277’, ’F50’, ’F218’, ’F67’, ’F9’, ’F54’, ’F220’, ’F75’, ’F320’, ’F70’, ’F226’, ’F86’, ’F59’, ’F108’, ’F84’, ’F296’, ’F222’
UDP-Lag (52)	’F254’, ’F301’, ’F231’, ’F104’, ’F110’, ’F279’, ’F83’, ’F129’, ’F255’, ’F84’, ’F322’, ’F111’, ’F108’, ’F76’, ’F135’, ’F127’, ’F302’, ’F43’, ’F79’, ’F272’, ’F250’, ’F248’, ’F217’, ’F222’, ’F128’, ’F67’, ’F223’, ’F82’, ’F71’, ’F9’, ’F226’, ’F130’, ’F296’, ’F225’, ’F78’, ’F319’, ’F277’, ’F323’, ’F230’, ’F253’, ’F320’, ’F303’, ’F219’, ’F218’, ’F220’, ’F274’, ’F300’, ’F77’, ’F22’, ’F74’, ’F75’, ’F10’
Web (26)	’F302’, ’F79’, ’F16’, ’F82’, ’F67’, ’F75’, ’F76’, ’F84’, ’F5’, ’F297’, ’F78’, ’F279’, ’F320’, ’F274’, ’F106’, ’F296’, ’F110’, ’F253’, ’F277’, ’F128’, ’F301’, ’F248’, ’F303’, ’F222’, ’F223’, ’F43’

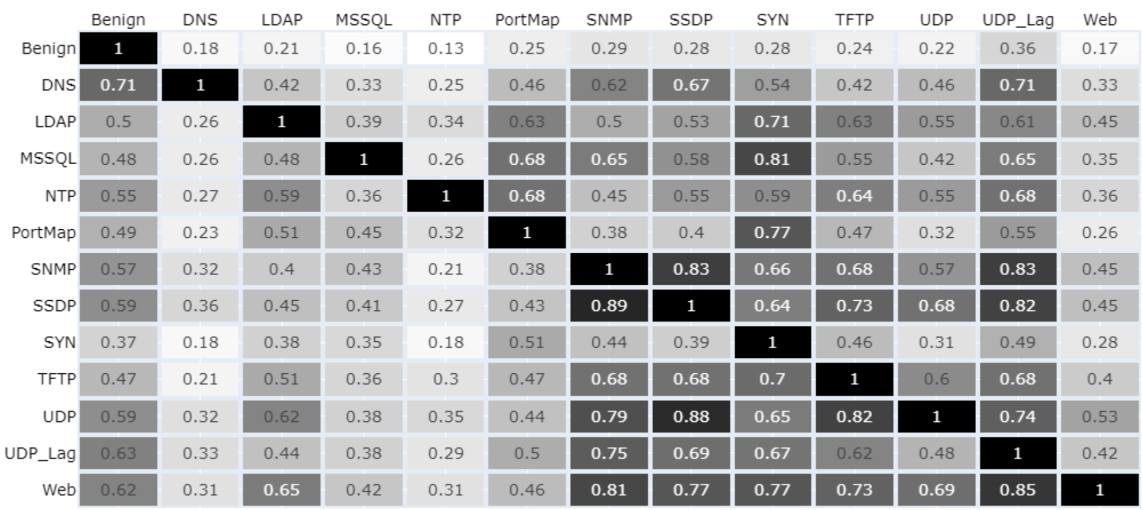

FIGURE 9. Pair intersections between the explanations of TP samples. Each cell in the row is the ratio of the number of intersected features to the number of features of the row label.

TABLE 6. The features\nibbles of each TP class explanation that have no intersection (unique features) with other explanations.

Class (#)	Features Name
Benign (52)	'F99', 'F207', 'F96', 'F100', 'F66', 'F68', 'F40', 'F318', 'F233', 'F64', 'F241', 'F172', 'F143', 'F186', 'F140', 'F148', 'F158', 'F120', 'F203', 'F189', 'F286', 'F315', 'F168', 'F90', 'F240', 'F192', 'F80', 'F177', 'F193', 'F243', 'F181', 'F72', 'F288', 'F149', 'F124', 'F175', 'F210', 'F185', 'F204', 'F278', 'F132', 'F122', 'F212', 'F242', 'F282', 'F244', 'F290', 'F123', 'F234', 'F190', 'F184', 'F284', 'F134', 'F146'
DNS (1)	'F73'
LDAP (2)	'F20', 'F247'
MSSQL (1)	'F44'
NTP (1)	'F221'
PortMap (4)	'F36', 'F34', 'F37', 'F6'
SNMP (0)	-
SSDP (1)	'F39'
SYN (11)	'F103', 'F49', 'F61', 'F53', 'F11', 'F299', 'F23', 'F271', 'F55', 'F52', 'F298'
TFTP (1)	'F47'
UDP (0)	-
UDP-Lag (0)	-
Web (0)	-

IPver	IHL	TOS		TotLen			
F0	F1	F2	F3	F4	F5	F6	F7
Ipid				Flags	FragOffs		
F8	F9	F10	F11	F12	F13	F14	F15
TTL		TrProt		HeadChs			
F16	F17	F18	F19	F20	F21	F22	F23
SrcAdd							
F24	F25	F26	F27	F28	F29	F30	F31
DstAdd							
F32	F33	F34	F35	F36	F37	F38	F39
SrcPort				DstPort			
F40	F41	F42	F43	F44	F45	F46	F47
SeqNum							
F48	F49	F50	F51	F52	F53	F54	F55
AckNum							
F56	F57	F58	F59	F60	F61	F62	F63
Offset	Reserved	TCPflgs		Window			
F64	F65	F66	F67	F68	F69	F70	F71
Chksum				UrPnter			
F72	F73	F74	F75	F76	F77	F78	F79

(a)

IPver	IHL	TOS		TotLen			
F0	F1	F2	F3	F4	F5	F6	F7
Ipid				Flags	FragOffs		
F8	F9	F10	F11	F12	F13	F14	F15
TTL		TrProt		HeadChs			
F16	F17	F18	F19	F20	F21	F22	F23
SrcAdd							
F24	F25	F26	F27	F28	F29	F30	F31
DstAdd							
F32	F33	F34	F35	F36	F37	F38	F39
SrcPort				DstPort			
F40	F41	F42	F43	F44	F45	F46	F47
UDPLen				Chksum			
F48	F49	F50	F51	F52	F53	F54	F55

(b)

FIGURE 10. Mapping the nibbles to their corresponding header fields in the transport layer. (a) TCP/IP header and (b) UDP/IP header.

its results to those obtained from explaining the LSTM model using the original SHAP.

B. EXPERIMENTAL RESULTS

In this subsection, we can summarize our experimental results in three folds: 1) the classification of the DDoS attacks in the raw CICDDoS2019 dataset using the LSTM model and calculating the classification performance. 2) the explanation of the LSTM predictions using the SHAPP and the original SHAP, mapping the resulted explanations into their corresponding packet fields, and evaluation the SHAPPD approach by comparing its explanations to those are obtained from the original SHAP. 3) the explanation of FP predictions of the LSTM model using the SHAPPD approach and justification of these predictions.

1) DDOS ATTACK CLASSIFICATION

We use the LSTM model designed in Figure 6 to conduct multiclass classification on the preprocessed raw traffic samples mentioned in Section III. We classify 12 classes of DDoS attacks extracted from CICDDoS2019 plus to the benign class. Table 4 shows that the LSTM model provides high performance on these classes where the metrics (accuracy, precision, recall, and F1-score) have very high values while the metrics FPR and FNR have very low values.

Figure 7 shows the confusion matrix of the multiclass classification on the CICDDoS2019 test set. The diagonal of the confusion matrix represents the true positive (TP) samples while the columns except the diagonal cells represent the false positive (FP) samples. We note from Figure 7 that the number of FP is few and this indicates a good performance of the classification model. However, The model can misclassify some samples of the dataset, which will cause the FPs. We can notice from the figure that the highest number of FPs are 320 for being misclassified as Benign, 105 for being misclassified as Web, and 100 for being misclassified as DNS.Such cases will be investigated by our proposed approach in the following sections.

2) LSTM MODEL EXPLANATION

In this subsection, we employ the SHAPPD approach to explain the predictions of the LSTM model. We follow the procedures in Algorithm 1 to explain TP and FP predictions of the LSTM classification on 13 network traffic classes extracted from the CICDDoS2019 dataset.

TABLE 7. The result of mapping SHAPPD explanation of each attack in the CICDDoS2019 into their corresponding TCP/UDP header fields of the transport layer.

Fields	Benign	DNS	LDAP	MSSQL	NTP	PortMap	SNMP	SSDP	SYN	TFTP	UDP	UDP-Lag	Web
IPver	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
IHL	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
TOS	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
TotLen	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓
IPid	✓	✗	✗	✗	✗	✓	✓	✓	✓	✗	✓	✓	✗
Flags	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
FragOffs	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
TTL	✗	✗	✓	✓	✓	✓	✓	✓	✗	✓	✓	✗	✓
TrProt	✗	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
HeadChs	✗	✗	✓	✓	✗	✓	✗	✓	✓	✓	✗	✓	✗
SrcAdd	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
DstAdd	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗	✗	✗	✗
SrcPort	✓	✗	✗	✗	✗	✗	✓	✓	✗	✓	✗	✓	✓
DstPort	✗	✗	✗	✓	✗	✗	✓	✗	✗	✓	✗	✗	✗
UDPLen	✗	✗	✓	✗	✗	✗	✓	✗	✗	✓	✓	✗	✗
SeqNum	✓	✗	✗	✗	✗	✓	✗	✗	✓	✗	✗	✗	✗
AckNum	✗	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Offset	✓	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
Reserved	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗
TCPflgs	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓
Window	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✗
Chksum	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✓	✗	✓
UrPnter	✓	✗	✗	✗	✗	✗	✗	✗	✓	✗	✗	✗	✓

TABLE 8. The result of mapping SHAPPD explanation of each attack in the CICDDoS2019 dataset into their corresponding header fields of the application layer.

Class	Fields	✗|✓	Class	Fields	✗|✓	Class	Fields	✗|✓
DNS	Id	✗	LDAP	MesgID	✓	MSSQL	PktType	✓
	DNSflgs	✗		ProtOP	✓		LstPktInd	✗
	QDcount	✗		-	-		PktSize	✗
	NScount	✗		-	-		Unkwn	✗
	ARcount	✓		-	-		-	-
NTP	NTPflgs	✗	PortMap	FrgHder	✓	SNMP	SNMPver	✓
	Stratum	✓		xid	✗		Commy	✗
	Poll	✗		MesgTy	✗		-	-
	Precision	✗		PRCver	✓		-	-
	RoDely	✗		RPCprog	✗		-	-
	RoDisp	✓		ProgVer	✗		-	-
	RefID	✓		Proced	✓		-	-
	RefTims	✗		Creden	✗		-	-
	OrgTims	✗		Verifier	✗		-	-
	RecTims	✓		PMprog	✗		-	-
	TranTims	✗		PMver	✗		-	-
	-	-		PMprot	✗		-	-
SSDP	Search	✓	TFTP	OPcode	✓	Web	StrtLine	✓
	Host	✓		Block #	✓		Host	✗
	Man	✗		-	-		Conn	✗
	-	-		-	-		AccEncod	✓
	-	-		-	-		Accept	✓

a: TRUE POSITIVE (TP) EXPLANATIONS

Figure 8 presents explanations of true positive predictions of all 13 classes in the raw CICDDoS2019 dataset. All the 324 features/nipples used in the data classification are presented in Figure 8 as (18 $\times$ 18) matrix starting from F0 into F324. The shaded features in this figure represent the class explanation S that is provided by the SHAPPD approach illustrated in Algorithm 1.

The explanation features of each class in Figure 8 are listed in Table 5 in descending order depending on their Shapley values. For example, the benign class has 92 features in its explanation starting with F99 which has the largest Shapley value, and ending with F146 which has the lowest Shapley value. The feature with the large Shapley value has more contribution to the class prediction.

The pair intersection between the explanations is shown in Figure 10. Each cell in this figure contains the ratio of the number of common features to the number of features in the intended class. As an example, consider the first row of Figure 10. In this case, the intended class is "Benign," which has 92 features within its explanation. Out of these features, 17 intersect with the "DNS" class, which itself has 24 features in its explanation. Therefore, the ratio will be $^{17}/_{92} = 0.18$ as shown in the second cell of the first row. We can see from Figure 10 that the largest ratio is 0.89 of SNMP $\cap$ SSDP to SSDP ($^{39}/_{44}$) followed by 0.88 of SSDP $\cap$ UDP to UDP ($^{30}/_{34}$). The common features between the pair explanations indicate the presence of a degree of similarity in the behavior of the corresponding DDoS classes. This degree of similarity is shown as similar features (nibbles) between the DDoS attacks but sometimes is not enough to produce similar predictions of the classification model.

On the other hand, the classes that exhibit different behavior reflect different features in their explanations. We can list in Table 6 the unique features of each class explanation. These features appear only in the explanation of the intended class, and they do not show any intersection with other classes. We can note from the result in this table that the benign class includes several unique features in its explanation compared to the other DDoS classes. This is expected because of the difference in the behavior between the benign and the DDoS attacks and the similarity in the behavior of DDoS attacks. The classes SNMP, UDP, UDP-Lag, and Web do not have any unique features while the rest of the DDoS classes have a few unique features compared to the Benign class.

b: GROUPING THE EXPLANATIONS

We use a nibble (4 bits) as raw dataset because it represents the smallest structure unit in protocol header fields. For example, the Header Length (IHL) field in the IP header consists of one nibble while the Time to Live (TTL) field consists of 2 nibbles and the Identification field consists of 4 nibbles. Consequently, the explanation features are also represented by nibbles. The deal with an explanation as individual nibbles often is meaningless, particularly for the header fields consisting of more than one nibble. To address this issue, we adopt a grouping process to put the related nibbles together to reconstruct the packet header fields.

Figure 10 illustrates the mapping process between the nibbles and the protocol header fields. For example, nibbles F2 and F3 are assigned to the Type of Service (TOS) field. Subplot (a) of Figure 10 represents packet header fields of TCP/IP protocols, whereas Subplot (b) represents packet header fields of UDP/IP protocols. The first 40 nibbles (F0 - F39) with the orange color represent the IP header fields, which are common between Subplots (a) and (b). The rest of the nibbles (F40 - F79) in (a) represent TCP header fields whereas the rest of the nibbles (F40 - F55) in (b) represent UDP header fields. We use Figure 10 to assign the class explanation to header fields of TCP and UDP protocols. The type of transport layer protocol (TCP or UDP) in the dataset can be identified from the protocol number field (TrProt) in the IP header. This field takes the hexadecimal value 0×06 for the TCP and 0×11 for the UDP.

After checking the TrProt field, we found that all the DDoS attacks in the CICDDoS2019 dataset were generated using UDP protocol except PortMap, SYN, and Web attacks were generated using TCP protocol. The benign traffic was generated using both TCP and UDP protocols.

Table 7 shows the result of mapping the explanations in Table 5 into TCP and UDP header fields. The first column lists all TCP and UDP header fields whereas the rest of the columns present the result of the mapping process for each class. It is worth mentioning that the Chksum field in the first column is common between the TCP and UDP headers.

The presence of the correct sign (✓) in the class column indicates that the corresponding fields have been derived from the mapping process of the class explanation. We notice from the table that several header fields are not assigned to any of the classes therefore all cells of its row include (✗) sign. These fields are IPver, IHL, TOS, Flags, FragOffs, SrcAdd, and Reserved. We can also see that the DstAdd field assigns (✗) for all classes except for the PortMap. This is because the PortMap class was generated in a victim network different from that of the other classes. The PortMap network includes a Web server with a different IP address. The reader can refer to [17] for more details. We can see from Table 7 that TotLen, TrProt, SeqNum, TCPflgs, and UrPnter fields were assigned

TABLE 9. The result of mapping SHAPPD explanation of each attack in the CICDDoS2019 dataset into their corresponding header fields of both the transport and application layers.

Class	Fields	Class	Fields	Class	Fields	Class	Fields
DNS	Chksum	LDAP	TTL	MSSQL	TTL	NTP	TTL
	ARcount		HeadChs		HeadChs		Stratum
	PL(20)		UDPLen		DstPort		RoDisp
	-		MesgID		Chksum		RefID
	-		ProtOP		PktType		RecTims
	-		PL(17)		PL(22)		PL(10)
PortMap	TotLen	SNMP	IPid	SSDP	IPid	SYN	TotLen
	IPid		TTL		TTL		IPid
	TTL		SrcPort		HeadChs		TrProt
	TrProt		DstPort		SrcPort		HeadChs
	HeadChs		UDPLen		Chksum		AckNum
	DstAdd		Chksum		Search		TCPflgs
	SeqNum		SNMPver		Host		Window
	FrgHder		PL(40)		PL(19)		UrPnter
	PRCver		-		-		PL(48)
	Proced		-		-		-
	PL(22)		-		-		-
TFTP	TTL	UDP	IPid	UDP-Lag	IPid	Web	TotLen
	HeadChs		TTL		HeadChs		TTL
	SrcPort		UDPLen		SrcPort		SrcPort
	DstPort		Chksum		PL(48)		TCPflgs
	UDPLen		PL(30)		-		Chksum
	OPcode		-		-		UrPnter
	Block #		-		-		StrtLine
	PL(38)		-		-		AccEncod
	-		-		-		Accept
	-		-		-		PL(6)

TABLE 10. The result of mapping the original SHAP explanation of each attack in the CICDDoS2019 dataset into their corresponding header fields of both the transport and application layers.

Class	Fields	Class	Fields	Class	Fields	Class	Fields
DNS	ARcount	LDAP	ProtOP	MSSQL	PktType	NTP	TTL
	PL (13)		PL (3)		PL(16)		RoDisp
	-		-		-		PL (7)
PortMap	UrPnter	SNMP	SNMPver	SSDP	IPid	SYN	SeqNum
	FrgHder		PL (14)		TTL		AckNum
	PL (16)		-		Search		UrPnter
	-		-		Host		-
	-		-		PL (15)		-
TFTP	TTL	UDP	IPid	UDP-Lag	PL (17)	Web	StrtLine
	OPcode		TTL		-		PL (4)
	Block #		PL (25)		-		-
	PL (37)		-		PL(48)		-

TABLE 11. The intersection between the explanations of TP and FP samples of the CICDDoS2019 classes using the SHAPPD approach.

Class	Benign	DNS	LDAPL	NTP	PortMap	SNMP	SSDP	SYN	UDP	UDP-Lag	Web
$\mathrm{EX_{TP}}$	92	24	38	22	47	47	44	71	34	52	26
$\mathrm{EX_{FP}}$	160	55	55	89	66	57	40	102	48	57	59
$\mathrm{EX_{TP}} \cap \mathrm{EX_{FP}}$	92	21	27	22	38	47	34	71	29	43	26
$(\frac{\mathrm{EX_{TP}} \cap \mathrm{EX_{FP}}}{\mathrm{EX_{TP}}})$ %	100	88	71	100	81	100	77	100	85	83	100

only for the classes (Benign, PortMap, SYN, and Web) that are generated based on the TCP protocol.

The impact of the application layer DDoS attacks on the behavior of network traffic is not limited to the header fields of the TCP/IP and UDP/1P packets. The header fields of the application layer protocols can be affected by the DDoS behavior and reflect it to the corresponding features in the DDoS class explanation. Therefore, we also map the explanation features into their corresponding header fields in the application protocols. By analyzing the raw traffic of the CICDDoS2019 dataset, we found that the DDoS attacks whose effect extends to the application layer are DNS, LDAP, MSSQL, NTP, PortMap, SNMP, SSDP, TFTP, and Web.

Table 8 shows the result of mapping the class explanation into the header fields of the DDoS attack protocols that work in the application layer. The 'Fields' columns state the header fields for each DDoS attack protocol in the application layer. The (✓) sign next the field indicates that this field has been extracted from the mapping process of the class explanation. All the header fields mentioned in Tables 7 and 8 are described in Appendix A.

The result in Table 9 summarizes the mapping process into both transport and application header fields of all the DDoS attack classes in the CICDDoS2019 dataset. Payload features (PL) for each class, in the table, refers to the rest of the features (nibbles) in the class explanation located in the range of the packet payload. For example, PL (20) in the DNS class means that the last 20 features of the DNS explanation are located in the packet payload. We note from Table 9 that some of the attack classes have common header fields, particularly in the network and transport layers. The SNMP and UDP classes have 4 common fields ('IPid', 'TTL', 'UDPLen', and 'Chksum') out of 7. The SSDP class also has 3 common fields ('IPid', 'HeadChs', and 'SrcPort') out of 7 with the UDP-Lag class and 4 common fields ('IPid', 'TTL', 'SrcPort', and 'Chksum') out of 10 with SNMP class. The result in Table 9 demonstrates that our proposed approach effectively provides a detailed analysis of the CICDDoS 2019 raw dataset. The approach could assign a specific set of header fields and a portion of the payload of the traffic packet that can differentiate the DDoS attacks in the analysis dataset.

c: COMPARISON SHAPPD TO ORIGINAL SHAP

To evaluate the SHAPPD approach, we compare its explanations of the LSTM model to those obtained from explaining the LSTM model using the original SHAP. In order to conduct the comparison, we will conduct the explanation using the original SHAP on the same results of the used LSTM model. Since the original SHAP deals with the input samples independently, the input vector is $(1 \times m)$ instead of an array $(T \times m)$. We then follow the same mapping steps to assign the resulting explanations to their packet header fields. Table 10 presents all the classes in the CICDDoS2019 dataset associated with their packet header fields and payload portion according to the resulting explanation using the original SHAP.

By comparing the results of the original SHAP in Table 10 to the results of the SHAPPD in Table 9, we can notice that the original SHAP provides less representative explanations of the series dataset (CICDDoS2019). The original SHAP, as shown in Table 10, failed to extract the explanations from all phases of the encapsulated packet traffic in several of the dataset classes. For example, the explanations of DNS, LDAP, MSSQL, SNMP, UDP-Lag, and Web don't include IP header fields compared to the SHAPPD explanations. Moreover, these explanations foremost concentrate on the packet features (PL). On the other hand, the results of our proposed approach (SHAPPD) in Table 9 provides more robust and representative explanations of all classes in the CICDDoS2019 dataset. These explanations can cover all phases of the encapsulated packet traffic and present several fields in each phase.

d: FALSE POSITIVE (FP) JUSTIFICATION

In this subsection, we perform the proposed explanation approach on the false positive (FP) samples. We can see from the confusion matrix in Figure 7 that there are few FP samples compared to the TP samples for each class. The number of FP samples can be found by summing the values of column cells of the confusion matrix except the cells at the matrix diagonal that represent the TP samples. Accordingly, the FP samples for all 13 classes are 320, 100, 4, 0, 2, 13, 26, 22, 1, 0, 49, 71, and 105 for Benign, DNS, LDAP, MSSQL, NTP, PortMap, SNMP, SSDP, SYN, TFTP, UDP, UDP-Lag, and Web, respectively. We follow the procedures in the SHAPPD algorithm to explain the LSTM predictions on the FP samples of the CICDDoS2019 classes to justify these predictions.

The FP of a certain class (i) is defined as the number of samples that are classified as i but they truly belong to other classes. The explanations of the FP samples of class (i) are expected to have common features with the explanations of TP samples for that class. Depending on these features, the model misclassifies the samples from other classes (not i) and predicts

TABLE 12. Description of the packet header fields in the transport and application layers.

Field	Discription
IPver	The Version field indicates the format of the internet header IPv4 or IPv6.
IHL	Internet Header Length is the length of the internet header in 32-bit words, and thus points to the beginning of the data.
TOS	The Type of Service provides an indication of the abstract parameters of the quality of service desired.
TotLen	Total Length is the length of the datagram, measured in octets, including internet header and data.
IPid	An identifying value assigned by the sender to aid in assembling the fragments of a datagram.
Flags	Various Control Flags: Reserved, DF, and MF.
FragOffs	Fragment Offset: This field indicates where in the datagram this fragment belongs.
TTL	Time to Live: This field indicates the maximum time the datagram is allowed to remain in the internet system.
TrProt	Protocol: This field indicates the next level protocol used in the data portion of the internet datagram.
HeadChs	Header Checksum: A checksum on the header only. Since some header fields change this is recomputed and verified at each point that the internet header is processed.
SrcAdd	Source Address: IP address identifies the specific host that transmitted the packet or frame onto the network.
DstAdd	Destination Address: IP address of the device to whom you want to send packet.
SrcPort	The source port number.
DstPort	The destination port number.
UDPLen	Length: is the length field in octets of this user datagram including this header and the data.
SeqNum	The sequence number of the first data octet in this segment.
AckNum	Acknowledgment Number: This field contains the value of the next sequence number the sender of the segment is expecting to receive.
Offset	This indicates where the data begins.
Reserved	A set of control bits reserved for future use.
TCPflgs	TCP Flags: The control bits: CWR, ECE, URG, ACK, PSH, RST, SYN, and FIN.
Window	The number of data octets beginning with the one indicated in the acknowledgment field that the sender of this segment is willing to accept.
Chksum	The checksum field is the 16-bit ones' complement of the ones' complement sum of all 16-bit words in the header and text. The checksum computation needs to ensure the 16-bit alignment of the data being summed.
UrPnter	Urgent Pointer: This field communicates the current value of the urgent pointer as a positive offset from the sequence number in this segment.
Id	A 16-bit identifier assigned by the program generates any kind of query. This identifier is copied the corresponding reply and can be used by the requester to match up replies to outstanding queries.
DNSflgs	Flags: QR, 0pcode, AA, TC, RD, RA, Z, and RCODE.
QDcount	An unsigned 16-bit integer specifying the number of entries in the question section.
ANcount	An unsigned 16-bit integer specifying the number of resource records in the answer section.
NScount	An unsigned 16-bit integer specifying the number of name server resource records in the authority records section.
ARcount	An unsigned 16-bit integer specifying the number of resource records in the additional records section.
MesgID	Message ID: A server identifies request packets sent by clients according to the message IDs and correctly returns response packets.
ProtOP	Protocol OP: Packet body, which carries packet type and authentication as well authorization information. Common packet types are bindRequest, bindResponse, searchRequest, searchResEntry, searchResDone, and searchResRef.
PktType	Packet Type: This field specifies the type of service packet such as TDS query, TDS login, and TDS authentication.
LstPktInd	Last Packet Indicator: This takes two values 0 and 1 for more received packets and for last packet, respectively.
PktSize	Packet Size: The size of packet in network byte order.
Unkwn	Unknown: It is always 0 and this has something to do with server-to-server communication/RPC stuff.
NTPflgs	Flags are LI: 2bits Leap Indicator, VN: 3bits Version Number, and Mode: 3bits.
Stratum	8-bit integer representing the stratum, with values: unspecified (0), primary server (1), secondary server (2 – 25), unsynchronized (16), and reserved (17 – 255).

TABLE 12. *(Continued.)* **Description of the packet header fields in the transport and application layers.**

Field	Discription
Poll	8-bit signed integer representing the maximum interval between successive messages, in log2 seconds.
Precision	8-bit signed integer representing the precision of the system clock, in log2 seconds.
RoDely	Root Delay: Total round-trip delay to the reference clock.
RoDisp	Root Dispersion: Total dispersion to the reference clock.
RefID	Reference ID: 32-bit code identifying the particular server or reference clock.
RefTims	Reference Timestamp: Time when the system clock was last set or corrected.
OrgTims	Origin Timestamp: Time at the client when the request departed for the server.
RecTims	Receive Timestamp: Time at the server when the request arrived from the client.
TranTims	Transmit Timestamp: Time at the server when the response left for the client.
SNMPver	SNMP version.
Commy	Community: An SNMP community string is a means of accessing statistics stored within a router or other device. Sometimes referred to simply as a community string or an SNMP string, it comprises the user credential—ID or password—delivered alongside a GET request.
Title	Indicates the SEARCH method to discover the services. M-SEARCH method uses the header format of HTTP/1.1.
Host	The host and port the message will be sent to. Typically, M-SEARC messages use the IP address 239.255.255.250 with the port number 1900.
Man	This defines the message type, for an M-SEARCH this will always be ssdp:discover.
Max	This specifies the maximum amount of seconds it takes for a device to respond.
OPcode	The code of operation includes Read request (1), Write request (2), Data (3), Acknowledgment (4), and Error (5).
Block #	The block numbers on data packets begin with one and increase by one for each new block of data.
FrgHder	The Fragment header is 32 bits that precede XDR packets over TCP. The most significant bit of the Fragment header indicates whether the packet is the last fragment, and the remaining 31 bits are the length of the Fragment that follows.
xid	All PortMap messages start with a transaction identifier, xid.
MesgTy	Message Type: RPC message protocol consists of two distinct structures: the call message and the reply message.
PRCver	RPC version: The version of Remote Procedure Call service.
RPCprog	RPC Program: The program used to map Remote Procedure Call service.
ProgVer	Program Version: The program version used in Remote Procedure Call service.
Proced	The procedure used in PortMap operation: NULL, SET, UNSET, GETPORT, DUMP, and CALLIT.
Creden	Credentials: This field is authentication field. There are two kinds of credentials: one in which the client uses its full network name, and one in which it uses its "nickname" given to it by the server.
Verifier hline PM-prog	This field is authentication field that the server generates in order to validate itself to the client. PortMap Program: The program used to map port based on the RPC service.
PMver	PortMap version: The version of PortMap service.
PMprot	PortMap Protocol: The used protocol in the napping process.
PMport	PortMap Port: The used port in the napping process.
StrtLine	Start line: The first line of the message which indicating what to do for a request or what happened for a response.
Host	The Host request-header field specifies the Internet host and port number of the resource being requested.
Conn	Connection: The Connection field allows the sender to specify options that are desired for that particular connection.
Accept	The Accept field can be used to specify certain media types which are acceptable for the response.
AccEncod	Accept-Encoding field is similar to Accept but restricts the content-coding that are acceptable in the response.

them as i. Table 11 presents the intersection between TP and FP explanations. The first row in Table 11 shows the data classes that include samples with FP prediction. We exclude MSSQL and TFTP classes because they do not have FP samples. The second row lists the number of features in explanation of TP samples while the second row lists the number of features in explanation of FP samples. The third row lists the number of common features between TP and FP explanations of each class. The rate of these common features to the number of features in TP explanations is listed in the last row.

We can see that the percentage of intersection between TP and FP explanations is high (100% for 5 classes and above 70% for the rest) of all classes. This indicates that the LSTM classification model made incorrect predictions for FP samples as these samples possess a majority of the features found in TP samples. This brings us to two separate conclusions; First, when the DDoS sample is incorrectly classified as one of the other DDoSs, which means that the classification model is unable to distinguish the DDoS classes that exhibit almost the same behavior. Second, when the DDoS sample is incorrectly classified as benign, which happens at the start points of the attack in the dataset where the few samples are labeled as background traffic (Benign) because of the difficulty in determining the specific starting time of the attack.

VI. CONCLUSION

In this paper, we propose an explanation approach SHAP with Pattern Dependency (SHAPPD) that evolves the original SHAP method to enhance its explanations. The SHAPPD considers the dependencies among the network traffic packets as well as the interdependencies between the features within the packet. We utilize the SHAPPD to explain the predictions of the LSTM model used to classify the DDoS attacks in the CICDDoS2019 dataset.

We first preprocess the CICDDoS2019 data by converting the PCAP data (in bits) into CSV data (in nibbles). For each packet in the dataset, we employ the first 324 nibbles to cover the TCP/IP or UDP/IP headers and portion of the payload. We then use the LSTM model to classify 13 different traffic classes in the CICDDoS2019 dataset. The classification's results demonstrate high values of performance metrics on all these classes.

After that, we use our proposed approach (SHAPPD) to explain the LSTM predictions of all the 13 classes in the CICDDoS2019 dataset. The output of SHAPPD is a class explanation that includes a set of nibbles/features that cause the class prediction. To provide an understanding of the set of nibbles inside the explanation set, we map these nibbles to their corresponding packet fields. The result of mapping provides a specific set of header fields and a portion of the payload of the traffic packet that can differentiate the DDoS attacks of the CICDDoS2019 dataset.

We evaluate our proposed approach (SHAPPD) by comparing its explanation results on the raw CICDDoS2019 dataset to those obtained from the original SHAP. The comparison results show that the SHAPPD can provide a more robust and representative explanation than the original SHAP.

We finally use the class explanation to justify the false positive (FP) predictions of the LSTM model. We compare the explanations of TP and FP to find the features that caused the LSTM to generate FP. The result shows that the percentage of intersection between TP and FP explanations is high for all classes. This indicates that the LSTM classification model made incorrect predictions for FP samples as these samples possess a majority of the features found in TP samples.

The approach (SHAPPD) is suitable for all the DL models that consider the dependencies of the dataset samples where the SHAPPD leverages these dependencies to enhance its explanation by providing more robust and representative explanations.

Our future work will include a study of implementing several recent DL models to classify the DDoS attacks and explaining these models using other explanation methods such as LIME and Anchor and show how adopting the dependency concept between the data samples can improve the explanation results.

APPENDIX A

see the Table 12.

REFERENCES

[1] J. Mirkovic and P. Reiher, "A taxonomy of DDoS attack and DDoS defense mechanisms," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 34, no. 2, pp. 39–53, Apr. 2004.
[2] K. Arora, K. Kumar, and M. Sachdeva, "Impact analysis of recent DDoS attacks," *Int. J. Comput. Sci. Eng.*, vol. 3, no. 2, pp. 877–883, Feb. 2011.
[3] M. M. Salim, S. Rathore, and J. H. Park, "Distributed denial of service attacks and its defenses in IoT: A survey," *J. Supercomput.*, vol. 76, no. 7, pp. 5320–5363, Jul. 2020.
[4] N. Hoque, D. K. Bhattacharyya, and J. K. Kalita, "Botnet in DDoS attacks: Trends and challenges," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 2242–2270, 4th Quart., 2015.
[5] V. D. M. Rios, P. R. M. Inácio, D. Magoni, and M. M. Freire, "Detection and mitigation of low-rate denial-of-service attacks: A survey," *IEEE Access*, vol. 10, pp. 76648–76668, 2022.
[6] S. T. Zargar, J. Joshi, and D. Tipper, "A survey of defense mechanisms against distributed denial of service (DDoS) flooding attacks," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 4, pp. 2046–2069, 4th Quart., 2013.
[7] M. Mittal, K. Kumar, and S. Behal, "Deep learning approaches for detecting DDoS attacks: A systematic review," *Soft Comput.*, vol. 27, no. 18, pp. 13039–13075, Sep. 2023.
[8] T. E. Ali, Y.-W. Chong, and S. Manickam, "Machine learning techniques to detect a DDoS attack in SDN: A systematic review," *Appl. Sci.*, vol. 13, no. 5, p. 3183, Mar. 2023.
[9] S.-Y. Ji, B. K. Jeong, and D. H. Jeong, "An analysis of temporal features in multivariate time series to forecast network events," *Appl. Sci.*, vol. 13, no. 18, p. 10411, Sep. 2023.
[10] H. Aydın, Z. Orman, and M. A. Aydın, "A long short-term memory (LSTM)-based distributed denial of service (DDoS) detection and defense system design in public cloud network environment," *Comput. Secur.*, vol. 118, Jul. 2022, Art. no. 102725.
[11] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, Jul. 2015, Art. no. e0130140.
[12] W. Guo, D. Mu, J. Xu, P. Su, G. Wang, and X. Xing, "LEMNA: Explaining deep learning based security applications," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2018, pp. 364–379.
[13] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why should I trust you?" in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Aug. 2016, pp. 1135–1144, doi: 10.1145/2939672.2939778.
[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 4768–4777.
[15] M. T. Ribeiro, S. Singh, and C. Guestrin, "Anchors: High-precision model-agnostic explanations," in *Proc. AAAI Conf. Artif. Intell.*, 2018, vol. 18, no. 1, pp. 1527–1535.

[16] R. Guidotti, A. Monreale, S. Ruggieri, D. Pedreschi, F. Turini, and F. Giannotti, ''Local rule-based explanations of black box decision systems,'' 2018, *arXiv:1805.10820*.

[17] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, ''Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy,'' in *Proc. Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2019, pp. 1–8.

[18] N. Namdev, S. Agrawal, and S. Silkari, ''Recent advancement in machine learning based Internet traffic classification,'' *Proc. Comput. Sci.*, vol. 60, pp. 784–791, Sep. 2015.

[19] H. Alhumyani, I. Alrube, S. Alsharif, A. Afifi, C. B. Amar, H. S. El-Sayed, and O. S. Faragallah, ''An efficient Internet traffic classification system using deep learning for IoT,'' *Comput., Mater. Continua*, vol. 71, no. 1, pp. 407–422, 2022.

[20] H. Yi and Z. Aye, ''Performance analysis of traffic classification with machine learning,'' *Int. J. Comput. Inf. Eng.*, vol. 15, no. 1, pp. 42–47, Feb. 2014.

[21] A. Azab, M. Khasawneh, S. Alrabaee, K.-K.-R. Choo, and M. Sarsour, ''Network traffic classification: Techniques, datasets, and challenges,'' *Digit. Commun. Netw.*, vol. 10, no. 3, pp. 676–692, Sep. 2022, doi: 10.1016/j.dcan.2022.09.009.

[22] S. Hwang, K. Cho, J. Kim, Y. Baek, J. Yun, and K. Han, ''Traffic classification approach based on support vector machine and statistic signature,'' in *Internet of Things, Smart Spaces, and Next Generation Networking*. Cham, Switzerland: Springer, 2013, pp. 332–339.

[23] J. J. Kponyo, J. O. Agyemang, G. S. Klogo, and J. O. Boateng, ''Lightweight and host-based denial of service (DoS) detection and defense mechanism for resource-constrained IoT devices,'' *Internet Things*, vol. 12, Dec. 2020, Art. no. 100319.

[24] B. Hu and Y. Shen, ''Machine learning based network traffic classification: A survey,'' *J. Inf. Comput. Sci.*, vol. 9, no. 11, pp. 3161–3170, Oct. 2012.

[25] Y. Zhang, X. Chen, L. Jin, X. Wang, and D. Guo, ''Network intrusion detection: Based on deep hierarchical network and original flow data,'' *IEEE Access*, vol. 7, pp. 37004–37016, 2019.

[26] R.-H. Hwang, M.-C. Peng, C.-W. Huang, P.-C. Lin, and V.-L. Nguyen, ''An unsupervised deep learning model for early network traffic anomaly detection,'' *IEEE Access*, vol. 8, pp. 30387–30399, 2020.

[27] P. Wang, F. Ye, X. Chen, and Y. Qian, ''Datanet: Deep learning based encrypted network traffic classification in SDN home gateway,'' *IEEE Access*, vol. 6, pp. 55380–55391, 2018.

[28] W. Wang, Y. Sheng, J. Wang, X. Zeng, X. Ye, Y. Huang, and M. Zhu, ''HAST-IDS: Learning hierarchical spatial–temporal features using deep neural networks to improve intrusion detection,'' *IEEE Access*, vol. 6, pp. 1792–1806, 2018.

[29] Y. Zeng, H. Gu, W. Wei, and Y. Guo, ''$Deep-Full-Range$: A deep learning based network encrypted traffic classification and intrusion detection framework,'' *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[30] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, ''Toward effective mobile encrypted traffic classification through deep learning,'' *Neurocomputing*, vol. 409, pp. 306–315, Oct. 2020.

[31] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, ''Deep packet: A novel approach for encrypted traffic classification using deep learning,'' *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, May 2019.

[32] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang, ''An LSTM-based deep learning approach for classifying malicious traffic at the packet level,'' *Appl. Sci.*, vol. 9, no. 16, p. 3414, Aug. 2019.

[33] R. K. Batchu and H. Seetha, ''An integrated approach explaining the detection of distributed denial of service attacks,'' *Comput. Netw.*, vol. 216, Oct. 2022, Art. no. 109269.

[34] T.-T.-H. Le, H. Kim, H. Kang, and H. Kim, ''Classification and explanation for intrusion detection system based on ensemble trees and SHAP method,'' *Sensors*, vol. 22, no. 3, p. 1154, Feb. 2022.

[35] M. Keshk, N. Koroniotis, N. Pham, N. Moustafa, B. Turnbull, and A. Y. Zomaya, ''An explainable deep learning-enabled intrusion detection framework in IoT networks,'' *Inf. Sci.*, vol. 639, Aug. 2023, Art. no. 119000.

[36] A. Warnecke, D. Arp, C. Wressnegger, and K. Rieck, ''Evaluating explanation methods for deep learning in security,'' in *Proc. IEEE Eur. Symp. Secur. Privacy*, Sep. 2020, pp. 158–174.

[37] M. Fan, W. Wei, X. Xie, Y. Liu, X. Guan, and T. Liu, ''Can we trust your explanations? Sanity checks for interpreters in Android malware analysis,'' *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 838–853, 2021, doi: 10.1109/TIFS.2020.3021924.

[38] L. S. Shapley, ''A value for n-Person games,'' in *The Shapley Value*. Princeton, NJ, USA: Princeton Univ. Press, 1988, pp. 31–40.

BASIL ASSADHAN (Senior Member, IEEE) received the M.Sc. degree in electrical and computer engineering from the University of Wisconsin and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University. He is currently an Associate Professor with the Electrical Engineering Department, King Saud University. His research interests include cybersecurity, network security, network traffic analysis, anomaly detection, and machine learning.

ABDULMUNEEM BASHAIWTH received the B.A. degree in electrical engineering from Hadhramout University and the M.Sc. degree in electrical engineering from King Saud University, where he is currently pursuing the Ph.D. degree with the Electrical Engineering Department. His research interests include cybersecurity, network security, network traffic analysis, anomaly detection, and machine learning.

HAMAD BINSALLEEH received the Master of Information Systems Security degree and the Ph.D. degree in computer science from Concordia University, Canada. He is currently an Assistant Professor with the Computer Science Department, Imam Mohammad Ibn Saud Islamic University. His current research interests include cybersecurity, network security, network traffic analysis and anomaly detection, passive DNS traffic analysis, malware analysis, and malware reverse engineering.