

Classifying Internet of Things (IoT) devices and non-IoT devices in data center edge network by one dimensional convolutional neural network using behavioral data

1st Sharath Ramamurthy

Upgrad Education Pvt. Ltd., Mumbai, India
Liverpool John Moores University, London, United Kingdom
sharu.murthy@gmail.com

2nd Dipankar Dutta

CSE Department, University Institute of Technology,
The University of Burdwan, Burdwan, India
dipankar2222@yahoo.com

Abstract—Now a days it is important to distinguish IoT and non-IoT devices from their network traffic in real-time for providing differential treatment to meet quality of service (QoS) requirements. This paper focuses on the classification of IoT and non-IoT devices based on their network traffic at edge data centers. We proposed a method using One-Dimensional Convolutional Neural Networks (1D-CNNs) which are simple, compact, and computationally less expensive for real-time processing needs. We used a dataset, which contain information from 22 IoT and 7 non-IoT devices that included both User Datagram Protocol (UDP) and Transport Control Protocol (TCP) flows. The classifier is constructed using 8 consecutive timesteps of payload exchanges carrying network payload data up to 96 bytes. Using 10-CV, the classifier achieves an accuracy of classification of about 98% on the test datasets.

Index Terms—IoT, Classification, 1D-CNN, Networking, Data Center, Edge Computing, Behavior Modeling

I. INTRODUCTION

Edge computing has increased the proximity of compute and cloud resources to data sources (IoT and non-IoT devices). This has allowed us to quickly process data produced by time-sensitive IoT devices (for example - data generated from a live camera of a driverless car). Edge data centers, which are situated closer to the devices, are one of the essential elements of the edge computing idea. Data transmission from several IoT and non-IoT devices is frequently aggregated by edge data center switches and routers. The consolidated traffic is sent to servers for additional processing via the data center network's compute and storage units. The network infrastructure providers (data center providers, internet service providers, etc.) are under increasing pressure to handle the network traffic that is being aggregated at their network as more and more IoT devices are connected to the internet. Therefore, it is crucial to distinguish between IoT and non-IoT devices from their network traffic at edge data centers in real-time so that network providers can give business-specific differential treatment. IoT and non-IoT devices connected to an edge data center are shown in Fig.1.

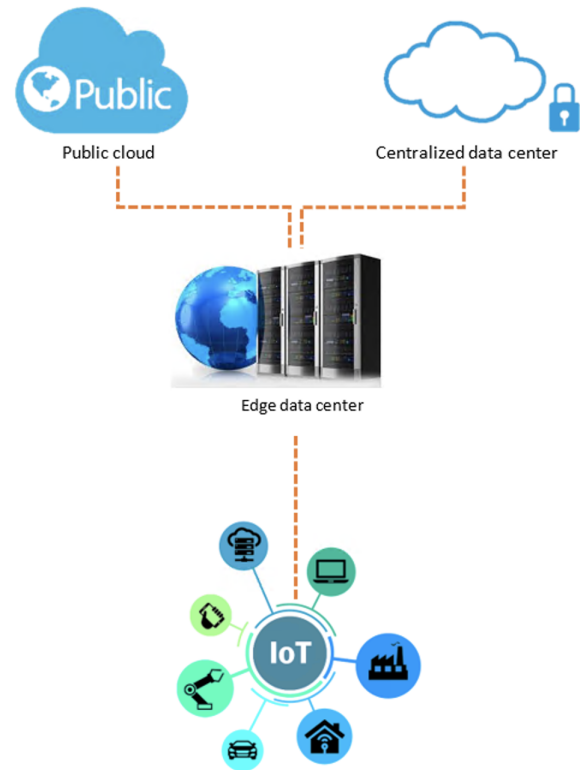


Fig. 1. Edge data center connecting IoT and non-IoT devices

Since the beginning of the internet, network traffic classification has been one of the most popular research areas. The majority of crucial network services and applications, including resource management, monitoring, security, accounting, etc., rely on this. It is one of the key components for long-term network resource planning since it allows for the

analysis of applications and resource expectations from network traffic, which allows for the prediction of future resource needs. Many network services, including firewalls, quality of service (QoS), intrusion detection systems (IDS), and others, benefit directly from network traffic classification in addition to resource management. As a result, one of the main areas of research in communication networks is classification. For the classification of network traffic in the past, deep packet inspection (DPI) and port-based approaches were used [1]. Application port numbers assigned by the Internet Assigned Numbers Authority (IANA), which are used in protocols like Internet Protocol (IP), User Datagram Protocol (UDP), and Transport Control Protocol (TCP), were the foundation of port-based techniques. The lack of standardized port numbers for all applications makes port-based techniques unreliable. Later, DPI-based techniques were used, which peek deeper into packets to identify the application. However, computational cost of these techniques are high, and they also fail when network traffic payloads are encrypted. Deep learning (DL) and machine learning (ML) algorithms have recently gained popularity and have demonstrated outstanding categorization accuracy. IoT device-generated network data has a consistent pattern and predictable behavior that can be used by ML models for identification and classification. According to [2]–[4], edge data center switches are traffic aggregators that combine traffic from both IoT and non-IoT devices. IoT device traffic, however, cannot be handled in the same manner as traffic from non-IoT devices. Depending on the business use cases, there could be different QoS (bandwidth, latency, etc.) needs, security requirements, and requirements for policy enforcement. In other words, understanding the different types of traffic will let network service providers for edge data centers implement network services including traffic priority for time-critical devices, intrusion detection, device monitoring, firewall management, etc. Edge data center switches must be able to distinguish between network traffic from IoT and non-IoT devices in real time for the classification framework to be useful and adaptable.

CNNs were initially created to address image classification issues [5]. Later, they were applied to a range of issues, including spatiotemporal models [6] and time series forecasting [7]. The convolution layer, pooling layer, and fully linked dense layer are the three different kinds of layers in a CNN architecture. Convolution layer is mostly made up of filters (kernel) to do feature mapping. The major factor in dimensionality reduction is the pooling layer. It aids in regularization by lowering model parameters. Fully connected layers are dense feed forward networks that are typically applied after convolutional layers to create nonlinear feature combinations and make predictions.

CNNs can be divided into one-dimensional (1D), two-dimensional (2D), and three-dimensional (3D) varieties depending on the type of input (1D/2D/3D). CNNs with more than 3D are uncommon. 1D-CNNs use 1D filters, which slide along 1D and are used in sequence-related tasks. 1D-CNNs use array multiplication instead of matrix multiplication, which is

more computationally efficient and more suited for real-time processing than 3D and 2D CNNs. Additionally, 1D-CNNs are being proposed for several real-time applications due to their straightforward and compact designs [6], [8].

The number of layers of CNN causes its computational complexity to rise. Our research aims to investigate computationally less expensive algorithms for accurately classifying network traffic from IoT and non-IoT devices. Because shallow CNNs are considered more suitable for real-time processing than deep CNNs, our work focuses on investigating shallow 1D-CNN models. In our work, the model is evaluated in terms of performance, accuracy, and other classification criteria.

II. RELATED WORK

There hasn't been much study done in the past on ML and DL approaches that classify IoT and non-IoT devices based on the communication behavior of the device. ML and DL models may be created to take advantage of the predictable behaviors and steady patterns of IoT traffic in order to identify and categorize IoT devices. Here, we include the known related research that have distinguished between IoT and non-IoT devices using behavior modeling techniques based on network traces. Behavioral models don't rely on a complex feature engineering stage, in contrast to non-behavioral models like statistical and network feature based. For the classification of IoT and non-IoT devices, these models employ the communication traffic that is directly provided (after any transformations and translations). We have included some similar research below that has classified IoT and non-IoT devices using behavioral approaches.

Two real-time categorization projects were put out in [9], [10], with a focus on the communication patterns of the devices. Without the use of costly feature engineering, these achieve device classification by feeding the classifier pre-processed packet data from network traffic. TCP traffic were the main focus of both research papers. Researchers employed the Stacked Long Short Term Memory (SLSTM) AutoEncoder [10]. Both a private dataset and a publicly accessible dataset were used by the authors. The public dataset was created using 21 different IoT devices, whereas the private dataset was created using 72 different IoT devices. The authors demonstrated that their model has a classification accuracy of 100% for previously observed IoT devices and a classification accuracy of 70% for previously unknown devices.

The TCP payload, which can contain up to 784 bytes, was given as a 28x28 vectored input to a neural network by Kotak and Elovici, who were able to categorize 10 different device classes with a 99.9% accuracy rate [9]. They used the dataset that was made available by A. Sivanathan et al. [11]. The experiment involved a short neural network with input, output, and one hidden fully connected layer. They only tested a portion of the dataset, and only a small number of IoT devices were employed in their tests.

1D-CNN has recently been utilized for categorization. 1D-CNNs have excelled in a variety of applications, as demon-

strated by [8]. Below are some examples of applications where 1D-CNNs have been tested and have shown promising results. These include automatic speech recognition (ASR) [12], real-time electrocardiogram (ECG) monitoring [13], and vibration-based structural damage detection in civil infrastructure [14]–[18].

In several applications, 1D-CNNs have produced positive outcomes in terms of accuracy and performance (computational cost). Because they are straightforward, compact, and computationally less expensive while yet being able to achieve high classification accuracy, 1D-CNN models are becoming more and more popular. As far as we are aware, there hasn't been any research that employed 1D-CNN to classify IoT and non-IoT devices based on behavioral data. The suggested classification problem is well suited for 1D-CNN models since they can detect non-axis parallel classification boundaries and capture sequences (flow of packets). Additionally, it may create classifiers that are incredibly accurate. Due to lower cost of shallow CNN than deep CNN, we used it for our research. Additionally, we have covered both the UDP and TCP flows that the devices exchange in our work.

III. RESEARCH METHODOLOGY

Our research's main objective is to create a classifier that can distinguish between IoT and non-IoT devices in network data in real time. This might be installed at edge data center switches, allowing network operators to manage the available network resources effectively and efficiently while accommodating the various traffic demands made by these devices.

A. Dataset

The University of New South Wales (UNSW), Sydney contributed the publicly accessible dataset that was used in this paper. The data set was gathered from a university-set up smart environment lab. The data set comprises of raw Packet Capture (PCAP) network traffic data that was acquired over a 20-day period from 29 distinct IoT and non-IoT devices. All of the devices that were used in the configuration are listed in Table I.

The dataset records the network traffic flow between devices (including control and user/application data). Over the course of 20 days, the dataset was gathered and organized on a daily basis. The device identifiers used as class labels in the presented data set are Media Access Control (MAC) addresses, which are specific to each device.

B. Preprocessing

The dataset is made up of 20 raw PCAP files that were used to capture the network traces of 28 devices over the course of 20 days. The raw PCAP files must be divided into PCAP files for each session that record the sequence of network packet exchanges between the device pairs during the course of the conversation. As the device pair participating in the communication is known from the network packet traces, this step aids in labeling the conversation. A session is a period of time during which a device pair can communicate in both

TABLE I
IoT AND NON-IoT DEVICES

IoT devices	Non-IoT devices
Insteon Camera	MacBook
TP-Link Smart plug	Android Phone
Samsung SmartCam	MacBook/iPhone
iHome	Samsung Galaxy Tab
Dropcam	Laptop
Belkin wemo motion sensor	iPhone
Withings Aura smart sleep sensor	
NEST Protect smoke alarm	
Amazon Echo	
Netatmo weather station	
Belkin Wemo switch	
Withings Smart scale	
Withings Smart Baby Monitor	
Blipcare Blood Pressure meter	
HP Printer	
Light Bulbs LiFX Smart Bulb	
Tribby Speaker	
TP-Link Day Night Cloud camera	
PIX-STAR Photo-frame	
Netatmo Welcome	
Nest Dropcam	
Smart Things	

directions. A combination of five tuples, including source IP, destination IP, transport protocol, (UDP/TCP) source, and destination port, uniquely identifies each session. To divide the PCAP files into sessions, we utilized the SplitCap tool [9]. The utility divides the provided PCAP file into several PCAP files, each of which records the sequence of network packet exchanges between the device pairs over the course of the conversation.

A sequence of packet payloads collected from each packet in the sequence for a certain flow (for each direction) from each device makes up the dataset for behavior modeling. An illustration is Fig. 2. One of these 4 byte application-specific data exchanged by an IoT device and enclosed in a TCP packet of a flow is shown in Fig. 2. A flow is the exchange of packets between two communicating devices in both the forward and backward directions. All of the control and user data packets transferred between devices are included in the per-session PCAP files recovered by the SplitCap program.

C. Cleaning

Below mentioned control flows can be seen in the per-session PCAP files retrieved using the SplitCap utility.

- Multicast and broadcast related communication exchanges
- Network traffic exchanged by router or gateway itself
- ARP, PING and other control network traffic

Control flows don't offer any information relevant to the classification job, hence they are irrelevant for the study. Therefore, these are taken out of the dataset.

D. Pre-processing packet payload data

Let packet PF_i^D and PB_j^D be the packet transmitted (forward) and received (backward) by the device D at time in-

> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface unknown, id 0
 > Ethernet II, Src: Invoxia_02:20:44 (18:b7:9e:02:20:44), Dst: Tp-LinkT_51:33:ea (14:cc:20:51:33:ea)
 > Internet Protocol Version 4, Src: 192.168.1.120, Dst: 46.105.38.79
 > Transmission Control Protocol, Src Port: 40234, Dst Port: 5228, Seq: 1, Ack: 1, Len: 4
 ▼ Data (4 bytes)
 Data: 0d0a0d0a
 [Length: 4]

0000	14	cc	20	51	33	ea	18	b7	9e	02	20	44	08	00	45	00	..	Q3	D	..	E	..
0010	00	38	84	92	40	00	40	06	9f	55	c0	a8	01	78	2e	69	..	8	..	@	..	@	..	
0020	26	4f	9d	2a	14	6c	7a	d4	22	8e	e9	eb	5f	5f	80	18	..	80	..	*	..	1z	..	
0030	ff	50	0b	3a	00	00	01	01	08	0a	00	02	d4	1b	f2	e1	..	P	..	:	
0040	db	f5	0d	0a	0d	0a											

Fig. 2. RAW PCAP data of a single packet

tervals i and j respectively in sequence. Forward and backward flows are specified in equation 1 and equation 2 respectively.

$$FlowForward^D = PF_1^D, PF_2^D, PF_i^D, \dots, PF_x^D \quad (1)$$

$$FlowBackward^D = PB_1^D, PB_2^D, PF_j^D, \dots, PF_y^D \quad (2)$$

The 1D-CNN model is fed the packet payload retrieved from each packet PF_i^D and/or PB_j^D for the entire sequence of flow from each device D . Input data is made up of K packet sequences from a flow. Each packet contains N data bytes, forming a $K \times N$ data. If the number of packets is less than K , then remaining sequences are padded with zero. The remaining data bytes would be padded with zero in the event that any packet contained less data bytes than N . Fig. 3 depicts the payload extraction procedure for $K = 4$ and $N = 50$.

Bytes make up the packet payload data unit. The range of byte values is 0 to 255. We divided each integer by 255 to convert those in the 0 to 1 range.

E. 1D-CNN Model

Despite the fact that there are numerous classifiers [19], CNN provides greater classification accuracy [20], in large part because of its automatic feature extraction skills and ability to recognize non-linear classification boundaries. We selected CNN for our job for these reasons. Our main objective is to investigate a shallow 1D-CNN model for real-time classification. A shallow 1D-CNN model is created during the model building phase. Using the flow traces of the devices' pre-processed packet data, a 1D-CNN model is trained and assessed. To achieve better results, the model is fine-tuned by modifying various hyper-parameters (such as filter sizes, number of convolution layers, and so on).

Fig. 4 depicts the architecture of the shallow 1D-CNN model employed in this work. The model is composed of two convolutional layers [21], two pooling layers [22], and a dense layer [21], as shown in the Fig. 4. Finally, classification is performed using an output layer with the SoftMax function [23].

Rectified Linear Unit (ReLU) [24] was utilized in 1D-CNN as the activation function for the convolutional and dense layers. In the first convolutional layer, second convolutional layer, first max pooling layer, second max pooling layer, and dense layer, there are 768, 800, 800, 800, and 64 neurons, respectively. The number of classes or devices is equal to the number of neurons in the output layer (here it is 23, 22 IoT devices and 1 for non-IoT device). As our goal function, we employed categorical cross-entropy loss [25]. In our investigations, stochastic optimization is carried out using the Adam optimizer, [26]. Adam is an adaptive learning rate optimizer created exclusively for neural network training. The Adam optimizer uses less memory and has a low computational cost. Ten epochs were discovered to be the most effective out of all the experiments, which used epochs ranging from five to twenty.

The following hyper parameters were tested and tuned for the best outcomes.

- Number of convolution/pooling layers
- Kernel/filter size
- Number of kernels/filters
- Number of filters
- Pooling size
- Batch size

Table II lists the values of hyperparameters

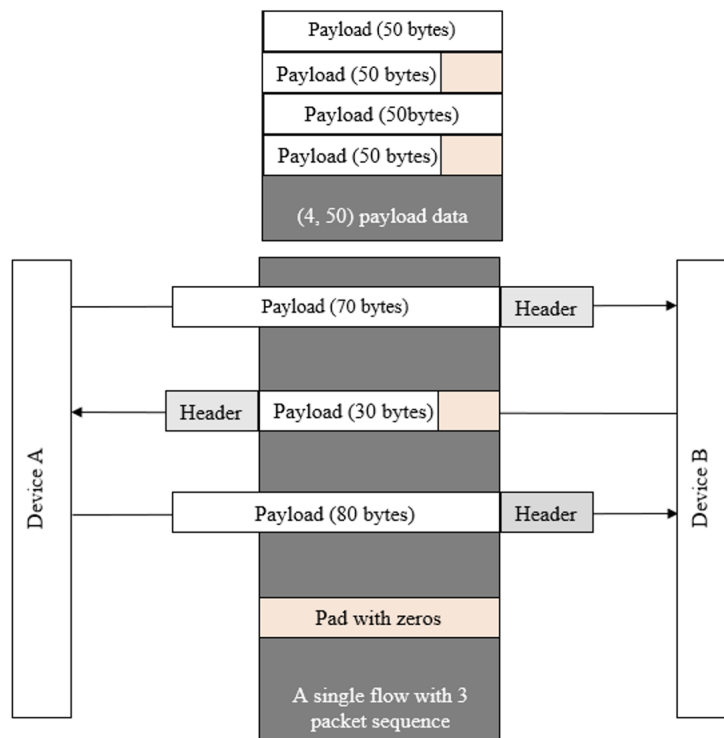
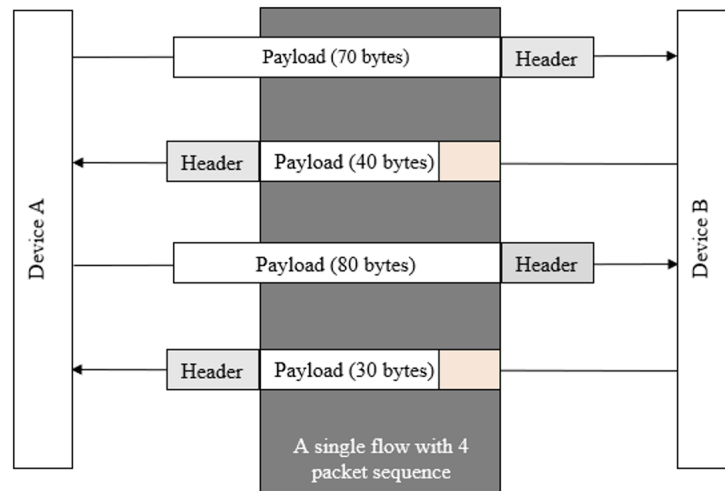
IV. MODEL EVALUATION

Different classification metrics that are available in the literature to comprehend the success of the categorization and its trade-offs with regard to real-time deployments, the 1D-CNN model is assessed. Some of these are Classification Accuracy, Precision, and True Positive Rate (TPR or Recall) [27], KAPPA [28], and F-Measure [29]. Among these we used Classification Accuracy here.

V. MODEL RESULTS

Table III contains classification results for each metric. For our experiment, we employed 10-CV [30] with an epoch of 10.

The findings indicate that the classifier created by 1D-CNN employing behavioral data is capable of accurately learning the



Payload (50 bytes)
Payload (50 bytes)
Payload (50bytes)
Pad with zeros
(4, 50) payload data

Fig. 3. Network flow payload data preparation with K=4 and N=50

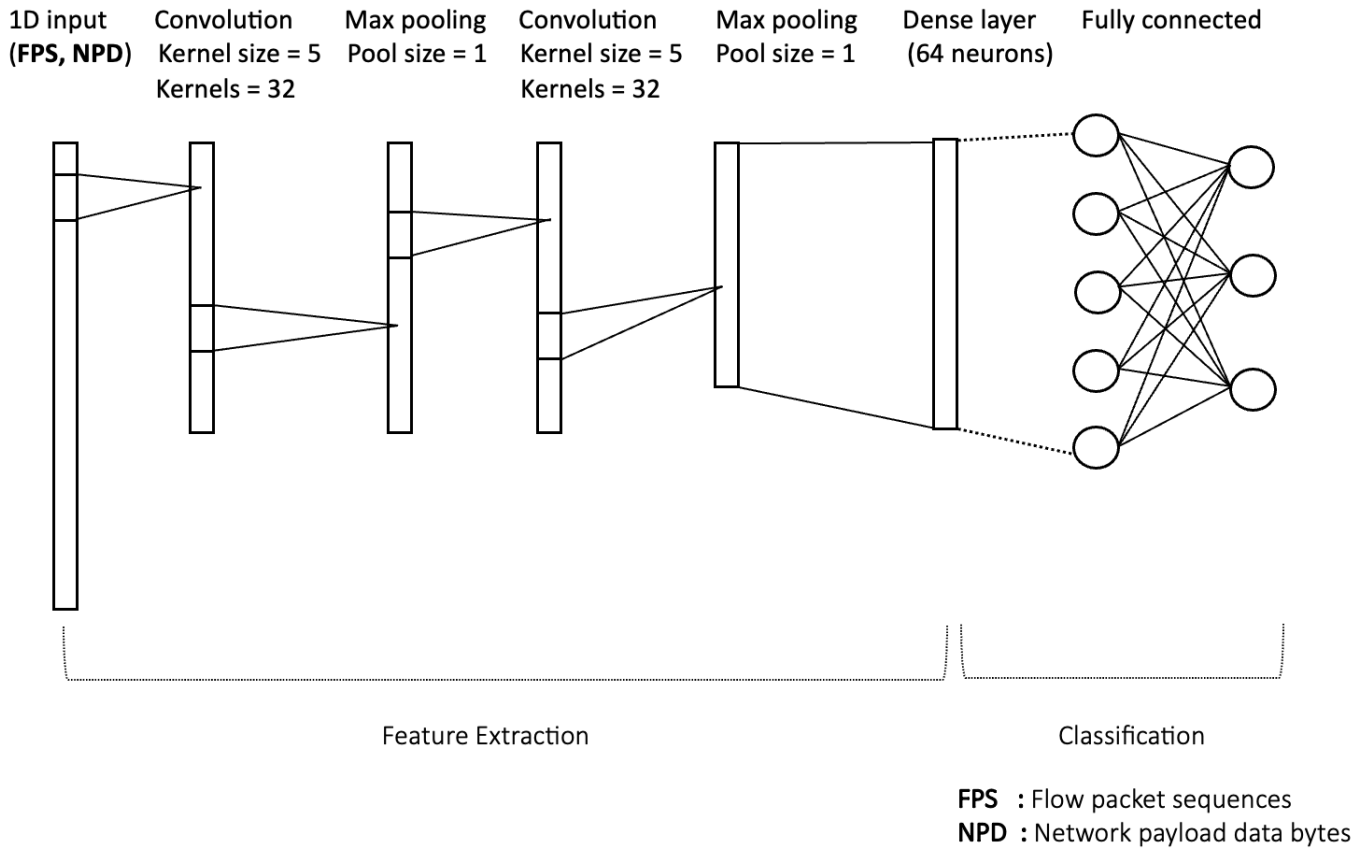


Fig. 4. the proposed shallow 1D-CNN

communication behaviors of devices. The proposed classifier is able to achieve 98% accuracy on the test dataset with as few as 8 consecutive timesteps of payload exchanges (forward and backward direction combined), each carrying as few as 96 bytes of network payload data bytes. It should be emphasized that the shallower 1D-CNN based classifier has less network parameters, making it less difficult to train. Additionally, it is simpler to extract packets from routers and switches than it is to extract or derive statistical and network flow properties. The proposed work is suited for real-time categorization as a result of these. The classifier will only need 60 μ secs to classify a test record, as shown in the Table III.

A. Results comparison

Here, we compare the outcomes of our work to earlier papers in the field of relevant research. For our research, Sivanathan 2019 [11], 20 days' worth of data were used in our research. Here, we compare the findings of our research to those of relevant earlier studies that have made use of analogous datasets. The key similarities are highlighted in Table IV.

Ortiz and al al. were able to obtain 66% on a comparable dataset using only TCP sessions [10], however the focus of our study was on both UDP and TCP. By doing this, we achieved a categorization accuracy of 98%.

Using the same dataset, Kotak and Elovici were able to reach a 99% accuracy rate [9]. However, they limited their study to TCP sessions. Additionally, they had only used a portion of the dataset with 10 classes for their research. However, the whole 20 days of data were used in our study to classify 23 separate groups using both UDP and TCP. Therefore, it is impossible to compare accuracy on an apples-to-apples basis.

VI. CONCLUSION

For the purpose of learning the communication behaviors of the device, we created a shallow 1D-CNN model. The classification of IoT and non-IoT devices using TCP and UDP traffic flows was done using the model. The shallow 1D-CNN model worked effectively, reaching a high classification accuracy of 98%, according to the data. The findings further demonstrate that IoT and non-IoT can be distinguished with 98% accuracy using just two convolutional layers and a pooling layer in a 1D-CNN model fed with eight consecutive packet exchanges. The classifier can operate in real time because it takes only a few microseconds (60 μ secs) to categorize test records.

REFERENCES

- [1] H. Tahaei, F. Afifi, A. Asemi, F. Zaki, and N. B. Anuar, "The rise of traffic classification in IoT networks: A survey," *Journal of Network and Computer Applications*, vol. 154, p. 102538, 2020.

TABLE II
VALUES OF HYPERPARAMETERS

Hyperparameter	Value	Remarks
Input dimension - (packet sequences of a flow, number of packet payload data bytes)	(8, 96)	Tests have shown that input dimensions (8, 96) which should only be one dimensional and whose values don't match the before specified values of 4, 50 brought about more consistent and effective outcomes. Beyond (8, 96), there was a slight improvement in categorization accuracy. Therefore, (8, 96) was chosen as the input size to reduce complexity.
Convolution and pooling layers	Two layers	Studies have demonstrated that the performance of categorization is not much improved by the inclusion of a third layer. Consequently, a two-layered structure is adopted to reduce complexity.
Kernel/filter size	Five	Five was a good kernel size for classification accuracy, according to experiments. After five, there was little to no discernible improvement. The kernel size of five is hence fixed.
Number of kernels/filters	32	Testshave showed that there was little gain in classification accuracy with more kernels or filters than 32. A trade-off exists between increasing kernel size and classification accuracy. 32 kernels are selected in order to balance the tradeoff without significantly reducing the model's complexity.
Pooling size	One	In order to keep the convolution output, the pooling size was set to one. Accuracy decreased as the pool size increased beyond one. Given that the dataset employed had less feature dimensions, dimensionality reduction did not play a significant role in the reduction of model complexity. Pool sizes of one are used to strike a compromise between model complexity and accuracy.
Batch size	16	Batch number 16 demonstrated the best balance between classification accuracy and training time out of all the batch sizes tested. So, a constant batch size of 16 is used.

TABLE III
CLASSIFICATION REPORT

Metric	Score
Training accuracy	98.3%
Test accuracy	97.9%
KAPPA score	97.6%
Weighted average precision	97.9%
Weighted average recall	97.9%
Weighted average F1-score	97.9%
Classification time for single test record	60 µsecs

TABLE IV
RESULTS COMPARISON

Author(s)	Accuracy	Remarks
Ortiz et al., 2019 [10]	66%	Only TCP sessions were considered.
Kotak & Elovici, 2021 [9]	99%	Only TCP sessions were considered. Only subset of the dataset was used with 10 classes.

- [2] H. Chang, A. Hari, S. Mukherjee, and T. Lakshman, "Bringing the cloud to the edge," in *2014 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pp. 346–351, IEEE, 2014.
- [3] P. P. Ray, "A survey on internet of things architectures," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 3, pp. 291–319, 2018.
- [4] R. J. Alzahrani and A. Alzahrani, "Survey of traffic classification solution in IoT networks," *International Journal of Computer Applications*, vol. 183, pp. 37–45, 2021.
- [5] W. Rawat and Z. Wang, "Deep convolutional neural networks for image classification: A comprehensive review," *Neural computation*, vol. 29, no. 9, pp. 2352–2449, 2017.
- [6] F. U. Islam, G. Liu, J. Zhai, and W. Liu, "VoIP traffic detection in tunneled and anonymous networks using deep learning," *IEEE Access*, vol. 9, pp. 59783–59799, 2021.
- [7] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.
- [8] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1D convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107398, 2021.
- [9] J. Kotak and Y. Elovici, "IoT device identification using deep learning," in *13th International Conference on Computational Intelligence in Security for Information Systems (CISIS 2020) 12*, pp. 76–86, Springer, 2021.
- [10] J. Ortiz, C. Crawford, and F. Le, "DeviceMien: network device behavior modeling for identifying unknown IoT devices," in *Proceedings of the International Conference on Internet of Things Design and Implementation*, pp. 106–117, 2019.
- [11] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath, and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1745–1759, 2018.
- [12] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu, "Convolutional neural networks for speech recognition," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 10, no. 22, pp. 1533–1545, 2014.
- [13] S. Kiranyaz, T. Ince, R. Hamila, and M. Gabbouj, "Convolutional neural networks for patient-specific ECG classification," in *2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pp. 2608–2611, IEEE, 2015.
- [14] O. Abdeljaber, O. Avci, S. Kiranyaz, M. Gabbouj, and D. J. Inman, "Real-time vibration-based structural damage detection using one-

dimensional convolutional neural networks,” *Journal of Sound and Vibration*, vol. 388, pp. 154–170, 2017.

- [15] O. Abdeljaber, O. Avci, M. S. Kiranyaz, B. Boashash, H. Sodano, and D. J. Inman, “1-D CNNs for structural damage detection: Verification on a structural health monitoring benchmark data,” *Neurocomputing*, vol. 275, pp. 1308–1317, 2018.
- [16] O. Avci, O. Abdeljaber, S. Kiranyaz, B. Boashash, H. Sodano, and D. J. Inman, “Efficiency validation of one dimensional convolutional neural networks for structural damage detection using a SHM benchmark data,” in *Proceedings of 25th International Congress of Sound and Vibration (ICSV)*, pp. 4600–4607, 2018.
- [17] T. Ince, S. Kiranyaz, L. Eren, M. Askar, and M. Gabbouj, “Real-time motor fault detection by 1-D convolutional neural networks,” *IEEE Transactions on Industrial Electronics*, vol. 63, no. 11, pp. 7067–7075, 2016.
- [18] S. Kiranyaz, A. Gastli, L. Ben-Brahim, N. Al-Emadi, and M. Gabbouj, “Real-time fault detection and identification for MMC using 1-D convolutional neural networks,” *IEEE Transactions on Industrial Electronics*, vol. 66, no. 11, pp. 8760–8771, 2018.
- [19] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, “A comprehensive survey on machine learning for networking: evolution, applications and research opportunities,” *Journal of Internet Services and Applications*, vol. 9, no. 1, pp. 1–99, 2018.
- [20] S. Dara and P. Tumma, “Feature extraction by using deep learning: A survey,” in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, pp. 1795–1801, IEEE, 2018.
- [21] K. O’Shea and R. Nash, “An introduction to convolutional neural networks,” *arXiv preprint arXiv:1511.08458*, 2015.
- [22] M. Sun, Z. Song, X. Jiang, J. Pan, and Y. Pang, “Learning pooling for convolutional neural network,” *Neurocomputing*, vol. 224, pp. 96–104, 2017.
- [23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.
- [24] A. F. Agarap, “Deep learning using rectified linear units (ReLU),” *arXiv preprint arXiv:1803.08375*, 2018.
- [25] P.-T. De Boer, D. P. Kroese, S. Mannor, and R. Y. Rubinstein, “A tutorial on the cross-entropy method,” *Annals of operations research*, vol. 134, no. 1, pp. 19–67, 2005.
- [26] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.
- [27] T. Eftestøl, “Controlling true positive rate in ROC analysis,” in *2009 36th Annual Computers in Cardiology Conference (CinC)*, pp. 353–356, IEEE, 2009.
- [28] M. L. McHugh, “Interrater reliability: the kappa statistic,” *Biochemia medica*, vol. 22, no. 3, pp. 276–282, 2012.
- [29] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation,” in *Australasian joint conference on artificial intelligence*, pp. 1015–1021, Springer, 2006.
- [30] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, “The kin k-fold cross validation,” in *20th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, pp. 441–446, i6doc. com publ, 2012.