IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 25, 2022, accepted February 17, 2022, date of publication February 22, 2022, date of current version March 2, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3153331

A Deep Learning Approach for IoT Traffic Multi-Classification in a Smart-City Scenario

AROOSA HAMEED, JOHN VIOLOS, AND ARIS LEIVADEAS, (Senior Member, IEEE)
Department of Software and Information Technology Engineering, École de Technologie Supérieure, Montreal, QC H3C 1K3, Canada

Corresponding author: Aris Leivadeas (aris.leivadeas@etsmtl.ca)

This work was supported in part by the CHIST-ERA-18-SDCDN-003-DRUID-NET Project "eDge computing ResoUrse allocatIon for Dynamic NETworks (DRUID-NET)."

ABSTRACT As the number of Internet of Things (IoT) devices and applications increases, the capacity of the IoT access networks is considerably stressed. This can create significant performance bottlenecks in various layers of an end-to-end communication path, including the scheduling of the spectrum, the resource requirements for processing the IoT data at the Edge and/or Cloud, and the attainable delay for critical emergency scenarios. Thus, a proper classification or prediction of the time varying traffic characteristics of the IoT devices is required. However, this classification remains at large an open challenge. Most of the existing solutions are based on machine learning techniques, which nonetheless present high computational cost, whereas they are not considering the fine-grained flow characteristics of the traffic. To this end, this paper introduces the following four contributions. Firstly, we provide an extended feature set including, flow, packet and device level features to characterize the IoT devices in the context of a smart environment. Secondly, we propose a custom weighting based preprocessing algorithm to determine the importance of the data values. Thirdly, we present insights into traffic characteristics using feature selection and correlation mechanisms. Finally, we develop a two-stage learning algorithm and we demonstrate its ability to accurately categorize the IoT devices in two different datasets. The evaluation results show that the proposed learning framework achieves 99.9% accuracy for the first dataset and 99.8% accuracy for the second. Additionally, for the first dataset we achieve a precision and recall performance of 99.6% and 99.5%, while for the second dataset the precission and recall attained is of 99.6% and 99.7% respectively. These results show that our approach clearly outperforms other well-known machine learning methods. Hence, this work provides a useful model deployed in a realistic IoT scenario, where IoT traffic and devices' profiles are predicted and classified, while facilitating the data processing in the upper layers of an end-to-end communication model.

INDEX TERMS Deep learning, edge computing, Internet of Things, machine learning, neural networks, traffic classification.

I. INTRODUCTION

Internet of Things (IoT) allows tens of billion devices to be connected over the Internet. Nonetheless, the rapid increase of IoT devices has also resulted in a colossal increase of the data generated by IoT devices. Specifically, the total data has quadrupled in just five years from 145 ZB in 2015 to 600 ZB in 2020 [1]. Furthermore, IoT not only enables new applications, but introduces new types of devices as well. For example, in the context of a smart environment, thousands of non-traditional Internet devices are used including smart sensors, alarms, traffic lights, cameras, weather stations, etc. generating an unprecedented volume of data for a variety of smart applications such as healthcare, industrial control, transportation and so on. However, these IoT devices are usually of limited computational abilities [2] and cannot manipulate locally the data generated.

This often urges the offloading of computational hefty IoT tasks to a remote infrastructure, a process called task offloading [3]. Edge Computing [4] is a viable solution for the task offloading as it allows to offer the necessary networking and computational resources at the edge of the network enabling at the same time the real time processing of the IoT data. However, as explained in [5], it is extremely difficult to estimate the edge resources needed due to the fact that (i) the IoT data are randomly generated, as a consequence of the

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

different types of devices and their dynamic cycle activity; and (ii) when there is a large number of IoT devices, the total communication delay may be affected on account of the constrained nature of the IoT access networks.

Hence, the importance to predict the time varying characteristics of the IoT devices (such as activity patterns, signaling patterns etc.) becomes evident. Furthermore, the classification of similar devices facilitates the estimation of the generated workload and can better guarantee a specific level of Quality of Service (QoS). Therefore, by classifying the IoT devices into different categories, the prediction of traffic characteristics can be more efficiently done. Additionally, a more accurate prediction of the resource requirements at the IoT access network (i.e. spectrum) and Edge infrastructures (i.e. computational and communication resources), can be achieved.

However, such an IoT device classification, often called device fingerprinting [6], presents several challenges. In particular, the existing IoT classification techniques do not consider the fine-grained characterization of IoT traffic, while they suffer from high computational cost for the data extraction and processing, and are often affected by high dimensional data and complexity. Accordingly, in this paper, we propose a two-stage based deep learning architecture in order to classify the IoT devices by considering a fine-grained set of network characteristics (features). To do so, firstly, we propose a two-step preprocessing algorithm while employing a feature selection and prioritization technique for the feature set under consideration. Our approach, facilitates the distribution of the features in the two stages avoiding the high dimensionality and overfitting problems of the training data.

The novelty of this paper lies in proposing a very accurate but considerably more lightweighted approach than the existing ones. Furthermore, the feature selection and prioritazion along with the combination of a deep learning model creates a unique and innovative approach for the problem of the IoT device classification. The novelty of our approach is strengthened by the fact that it can be generalized and applied in different datasets without losing any accuracy. Thus, the reproducibility of the results and the stability of our approach in different IoT contexts fortify the originality introduced.

In particular, the major contributions and novelty of this paper can be summarized as follows:

1) In order to perform a classification of the IoT devices, we have suggested an extended feature set comprising of flow, device, and packet level features. This approach provides a fine grained characterization of the traffic flow with less computational complexity for the classification.
2) A two step preprocessing algorithm is proposed that assigns relevance weights to the nominal (representing the qualitative data with numeric codes) features and provides scaling of the dataset using a MinMaxScaler method.
3) A statistical feature selection technique is employed to select the features with regard to their contribution to the classification of IoT devices. Furthermore, an investigation of correlated features at each level is provided using the Pearson correlation coefficient.
4) A two stage learning framework is presented with 99.9% accuracy for the first dataset under consideration and 99.8% for the second one, which proves the generalization of our approach. To determine the IoT device classification, we compute the classes for certain nominal and multivalued attributes at learning stage 0 using logistic regression. Following, we perform the final classification for numerical and single-valued features at stage 1 using a multilayer perceptron (MLP) neural network. The MLP network takes as an input a feature subset at each time and classifies IoT devices in a context of a smart environment. Furthermore, to achieve the optimal or near optimal MLP architecture, a random search based keras tuner is employed.

The rest of the paper is structured as follows: Section II highlights the related work in traffic classification, covering the most important methods and technologies applied in the IoT traffic classification domain. Section III provides the system model and necessary preliminaries for comprehending the classification problem in the context of the IoT domain. Additionally, this Section covers the description of the feature sets, their statistical characteristics and feature correlation, information that is necessary for the domain of data analysis that our paper touches upon. Section IV presents the two-stage proposed learning framework for the IoT device classification problem. Section V explains the algorithmic form of proposed preprocessing and learning model along with their asymptotic analysis. Sections IV and V fall under the domains of deep learning, machine learning and problem complexity, presenting all the necessary technical details. Section VI provides the performance evaluation results for both datasets under consideration. The conclusions and the future directions of this work are presented in Section VII. Finally, Table 1 presents the set of abbreviations used in this paper.

II. RELATED WORK

For the IoT device classification, significant emphasis has been given into aggregated traffic models, fingerprinting, and machine learning based solutions. The aggregated traffic models resort to mathematical and statistical distribution-based methods, which involve several probability distributions and mathematical techniques like stochastic processes to model the traffic. Following, the fingerprinting methods are used to identify the IoT devices leveraging information from network traces in order to correlate datasets. In particular, this category of classification identifies a device using information from the network packets during the communication over the network.

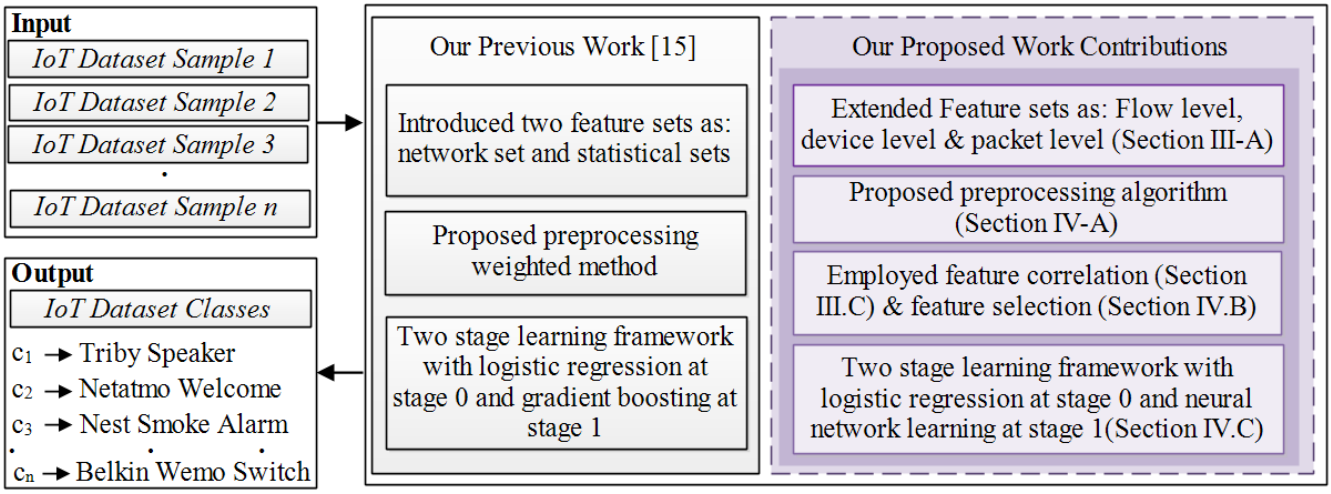


FIGURE 1. Overview of our previous work vs. proposed work contributions (shown in the purple boxes).

TABLE 1. List of abbreviations.

Abbreviations	Meaning
AdaGrad	Adaptive Gradient
Adam	Adaptive Moment Estimation
ANOVA	Analysis of Variance
BoW	Bag-of-Word
CMMPP	Coupled Markov Modulated Poisson Processes
DF	Do not Fragment Flag
DT	Decision Tree
FC	Fully Connected
FNR	False Negative Rate
FPR	False Positive Rate
GB	Gradient Boosting
IaT	Interarrival Time
IoT	Internet of Things
IP	Internet Protocol
KNN	K Nearest Neighbor
LR	Logistic Regression
LSTM	Long short-term memory
ML	Machine Learning
MLP-ANN	Multi-Layer Perceptron Artificial Neural Network
MSS	Maximum Segment Size
M2M	Machine-to-Machine
NB	Naive Bayes
NOP	No Option
OP	Output Layer
PDR	Packet Drop Rate
QoS	Quality of Service
RF	Random Forest
ReLU	Rectified Linear Units
RFE	Recursive Feature Elimination
SVM	Support Vector Machine
TTL	Time-to-Live
TCP	Transport Control Protocol
WS	Window Size
WSO	Window Size Option

Regarding the Machine Learning (ML) based schemes, they utilize existing algorithms to automatically learn complex patterns from the IoT traffic data. The algorithms used in these schemes are classified according to how the learning process is conducted. Four main classes are used to group ML algorithms: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. However, in the current literature, mostly supervised learning, unsupervised learning or a combination of these two are utilized in order to analyze, predict and model the IoT traffic and device characteristics.

With respect to aggregated traffic models, Laner *et al.* [7] proposed a Coupled Markov Modulated Poisson Processes (CMMPP) framework to capture the traffic behavior of a single machine-type communication along with the collective behavior of tens of thousands of devices. In [8] a classification strategy is designed for a fleet management use case incorporating three classes of M2M traffic states, namely periodic update, event-driven, and payload exchange. The authors in [9] proposed a model that estimates the M2M traffic volume generated in a wireless network-enabled connected home. However, the above works do not consider the fine-grained characterization of the IoT traffic, whereas the complexity of such methods grows linearly with the number of the devices. Furthermore, common communication patterns identified can be attributed to any sensing device under a specific use case (limitation 1).

There is also a significant effort to identify the type of the IoT devices using the fingerprinting method. For example, "IoT Sentinel" [10] is a classification system that can recognize and identify the IoT devices immediately after they are connected to a network using a single attribute vector with 276 network features. The "IoT Sentinel" framework can be further improved by extracting additional network features such as payload entropy, TCP payload length, and TCP window size [11]. Similarly, in [12] almost 300 network attributes are used from each TCP traffic session to classify the devices, using a majority voting for every 20 consecutive sessions.

The work in [13] utilized a deep learning approach in order to perform the device fingerprinting using the packet

TABLE 2. Comparison of related works.

Category	Ref.	Technology	Traffic source	Features
Aggregated Traffic Models	[7]	Coupled Markov Modulated Poisson Processes (CMMPP)	Simulated data for 30000 devices	No. of devices, distribution, time period
	[8]	Aggregated+SMM+CMMPP	Fleet management	time, packet size, direction, IP address, port no., APN
Fingerprinting	[10]	ML based model + SDN based traffic monitoring	Collected 27 IoT devices data	link protocol, network protocol, transport protocol, IP options, IP address, port numbers
	[11]	Device Fingerprinting+DT, GBM and Majority voting	Collected data of 14 IoT devices	Packet header features+ payload length, entropy, win.size
	[12]	Device white listing + RF	17 devices data	time to live statistical information
	[13]	4 DFP models+CNN	IPhone/ipad data	636 and 608 IAT graphs of Apple devices used for CNN
	[14]	Separate and label streams, examine traffic rate	Collected data from four smart devices	IP addresses, TCP ports, DNS queries
Machine Learning	[15]	KNN, DT, MLP, SVM	OS identification data	WS, TTL, DF, MSS, WSO, port no., NOP
	[16]	Single layer neural n/w	Traffic data from [23]	TCP payloads converted to grayscale images
	[17]	Regression approaches	Real time IoT application data	Node id, total messages transmitted and received, timestamps, success rate, latency, PDR, throughput
	[18]	Random Forest	Traffic data from [23]	Packet size, packet volume, IaT, duration, URG & PSH
	[19]	LSTM	Simulation data	Timestamp, bytes count, and the packets count
	[20]	RF, DT, SVM, KNN, NN and NB	Generate data from four IoT devices	packets sent, packets received, IaT between packets sent, IaT between packets received
	[21]	CNN+LSTM	RedIRIS dataset	source & dest. port, payload, WS, timestamp, direction
	[22]	GBM, RF, XGboost	Network traffic data	source IP, destination IP, port numbers
	[23]	NB+RF Classifier	Collected data from smart lab	port no., domain name, cipher suite, flow volume, duration, rate, DNS interval, NTP interval
	[24]	LR+GBM	Traffic data from [23]	IP and MAC addresses, port no., TTL, protocol, IaT, packet size, traffic rate, burstiness)

interarrival time. However, this approach is computationally intensive as all packet level information is utilized without any selection strategy. In [14], the traffic patterns of encrypted network flows are used to reveal the existence of a specific device inside a home network. However, obtaining such a great number of features require specialized hardware accelerators, thus resulting in high computational cost, longer classification duration and limited scalability due to the need of a deep packet inspection functionality (limitation 2).

Some related works also employed machine learning in order to perform traffic and device classification. Lippmann *et al.* [15] compared the K-nearest neighbor (KNN), Support Vector Machine (SVM), Decision Tree (DT) and Multilayer Perceptron (MLP), using the packet header information and concluded that KNN and DT provide better results. Kotak and Elovici [16] classified nine different device flows based on the device type using artificial neuron network. Regarding traffic classification, the authors in [17] predicted the QoS behavior of five different IoT applications in a smart building context, using several regression based ML approaches.

The work in [18] shows how to classify traffic and perform device identification using random forest. The list of key features used in the classification included the packet size, volume of packets, inter-arrival time, duration, urgent and push flags. Additionally, the authors in [19] performed a prediction of the IoT network traffic using Long Term Short Memory (LSTM). The features of dataset consisted of the timestamp, bytes count, and the packet count. A more comparative approach, was introduced in [20], where the authors presented a method to recognize the IoT devices using random forest, decision tree, SVM, k-nearest neighbors, simple neural network and naive bayes approaches.

Lopez-Martin *et al.* [21] classified the traffic applications using a multi-class neural network, which is proven to be effective in complex data structures. The authors in [22] proposed an individual binary classification model for each class in order to eliminate the complexity issue of multi-class classification. Sivanathan *et al.* [23] utilized the statistical attributes, signaling patterns and cipher suites along with machine learning for IoT device classification.

Nonetheless, these ML approaches are affected by the high data dimensionality, they are sensitive to the hyper-parameter tuning and they require a large number of training data. Moreover, the main constraint of the multi-class classification is scalability, as the high number of classes makes the classifier more complex and updating requires full retraining (limitation 3). A summary of the papers reviewed in this section is given in Table 2.

In our preliminary work [24], we tried to address some of these limitations by relying on typical machine learning techniques, such as logistic regression and gradient boosting. In this paper, we extend our preliminary framework to provide a more complete and detailed IoT multi-classification approach based on a deep learning solution. As this research is an extension of our previous study, we used the same IoT dataset [23]. However, in order to prove the generalization of our proposed methodology we also performed our experiments with a second IoT dataset [25]. Additionally, herein, we include a more extended feature set at three different levels such as: device, flow and packet.

This work also introduces a feature correlation mechanism, whereas specific features are selected for training models

which is not included in our previous work. Furthermore, for the new two stage learning framework, we apply an optimal searched neural network architecture at the second stage. Finally, a completely new performance evaluation section is presented. The particular section includes a new set of results for both datasets, new experiments, and additional comparisons with machine learning and deep learning approaches. The differences between our previous and proposed work are given in Fig. 1.

The extensions made in this paper are aligned in such a way to address the above cited limitations:

- To overcome limitation 1, we incorporate a fine-grained feature set at different network levels i.e., flow, device and packet level.
- To address limitation 2 and the high computational costs of complex features, we employ a statistical feature selection (i.e., ANOVA score) to select a subset of the available features at a time instance t.
- To address limitation 3, we propose a two-stage learning framework. Firstly, a relevance weighting-based preprocessing is performed for the available features, whereas different subsets of the selected features are utilized across these two stages to avoid the high dimensionality issue. Finally, the tuned hyperparameters are utilized in a neural network that achieves 99.9% accuracy for the first dataset and 99.8% for the second.

III. PROBLEM SETUP

In this section, we describe and formulate the IoT traffic classification problem, where different IoT devices are combined to their respective classes according to their distinctive characteristics. To help the reader follow the modeling of our work, Table 3 summarizes the key notation used throughout this paper.

In particular, a smart environment (e.g. smart city, home, grid, etc.) can be modeled as a network of S smart devices, generating M traffic flows. The devices are represented by the set $D = \{d_1, d_2.., d_s\}$, where d_s indicates the s^{th} smart device, where $1 \leq s \leq S$. Similarly, the set $T = \{t^1_{d_1}, t^2_{d_2}, \ldots, t^m_{d_s}\}$ represents the generated traffic flows, where $t^m_{d_s}$ denotes the m^{th} traffic flow in T generated by the s^{th} device, with $1 \leq m \leq M$ such that $M \subseteq S$. Furthermore, each traffic flow is constituted by a number of packets denoted by $P = \{p_{1m}, p_{2m}, \ldots, p_{km}\}$ where p_{km} represents the k^{th} packet of the m^{th} flow.

Regarding the features, the set F denotes the distinctive properties of the traffic flow $t^m_{d_s}$ which we want to classify. Each packet in P is a D-dimensional set of the network elements under consideration. These elements are represented as a feature space F, such that $F = \{f_1, f_2, f_3, .., f_i\}$, where f_i represents the i^{th} feature in the feature space F with $1 < i \leq 11$ (in this work we assume 11 distinctive features).

The set F consists of device, flow and packet level features, where f_1 represents the interarrival time, f_2 denotes the source IP address, f_3 is the destination IP address, f_4 shows the transport protocol used by each flow, f_5 is the source port number, f_6 denotes the destination port number, f_7 is the Time-to-Live (TTL) information, f_8 denotes the window size used by the transport layer, f_9 indicates the length of a packet, and f_{10} denotes the source Ethernet address, and f_{11} is the destination Ethernet address.

Furthermore, we assume that we have a given training set G, including pairs of input samples along with their class labels as $G = \{(x_1, c_1), (x_2, c_2), \ldots, (x_r, c_q)\}$. Accordingly, the set $C = \{c_1, c_2, \ldots, c_q\}$ denotes the available classes, where $c_q \in C$ represents the q^{th} class in C, while $C \subset D$ and $q \leq n$. Furthermore, $x_r \in X$ is the r^{th} input sample of the total set of samples $X = \{x_1, x_2, \ldots, x_r\}$, such that $X \subset P$ and $r \leq k$. Hence, the IoT Traffic Classification problem is defined as the task of estimating the class label c_q to the input vector x_r, where x_r belongs to a subset of a feature space F, $x_r \in X \subset F$. This task is accomplished using a classification

TABLE 3. Summary of the key notation.

Symbols	Meaning
D	Set of devices
S	Smart devices
M	Traffic flows
m^{th}	Last traffic flows
d_s	Last smart device in set S
T	Set of generated traffic flow
$t^m_{d_s}$	Last traffic flow generated by last device in set S
P	Set of packets generated by traffic flow
k^{th}	Last packet generated in the traffic flow
p_{km}	Last packet generated in the last flow
F	Set of features
f_i	Last feature in the feature set F
G	Set of training instances
X	Set of total sample instances
x_r	Last instance of features in set X
C	Set of classes or labels
c_q	Last class of features in set C
q^{th}	Last class
r^{th}	Any input sample of set X
D	Input vector of specific dimension
f_i	Feature in feature space
cov	Covariance between two features
σ	Standard deviation
ρ	Pearson's correlation coefficient
v	Feature vector
p	Probability for a combination of independent variables
β_0	Intercept
β_n	Regression coefficients
y_i	Dependent variable
x_n	Independent variable
l^{th}	Layer of neural network
$O^{(l)}_i$	Output of the i^{th} neuron at l^{th} layer
V	Nonlinear activation function
w	Weight of a neural network connection
B	Bias value applied at a layer
$O(n)$	Linear complexity
t	Number of training examples
e	Number of epochs
d	Number of neurons in each layer
$\cap$	Intersection between two sets
$\notin$	Not element of
$\Leftrightarrow$	Equivalent of

TABLE 4. Description of features in both datasets.

Level	Variable	Features	Description
Packet Level	f_1	IaT	Average time between two consecutive packet receptions
	f_9	Length	The length of network packet
Flow Level	f_2	IP.src	Source IP address
	f_3	IP.dst	Destination IP address
	f_4	Protocol	Protocol used by the flow
	f_5	Port.src	Port number of the client
	f_6	Port.dst	Port number of the server
	f_7	TTL	Maximum number of hops left for each packet to reach the destination
	f_8	WS	The amount of bytes the receiving device is capable to receive
Device Level	f_{10}	MAC.src	Source MAC address
	f_{11}	MAC.dst	Destination MAC address

rule or function $f(x) : X^D \rightarrow C$ that can predict the label C of unseen D dimensional input vector x_r.

A. FEATURE DESCRIPTION

As mentioned earlier, the available features can be categorized as follows:

1) DEVICE LEVEL FEATURES

In this category we consider the source and destination MAC addresses of the devices. Such features are extracted directly from the traffic traces. These features offer a characterization of the IoT traffic independent of the other two levels of features.

2) FLOW LEVEL FEATURES

This includes features such as source and destination IP addresses, protocol type of a flow, source and destination port numbers, the TTL information of a flow, and the window size used by the flow. This set can be used to extract the packet level features of a flow described below.

3) PACKET LEVEL FEATURES

This category includes the timestamp, the interarrival time (IaT), and the length of the packets. The interarrival time is the amount of time that elapses between a packet reception and the arrival of the one following it. As timestamp follows the normal (guassian) distribution, to calculate the interarrival time feature, we analyzed and extracted the time between the successive incoming traffic packets following a Gaussian's distribution with an average rate of 1 (since at each time unit one packet arrives). All of the above features along with their description are illustrated in Table 4. To prove the generality of our approach, we used the same feature sets for both datasets under consideration.

B. STATISTICAL CHARACTERISTICS OF THE FEATURES

Each feature f_i in the feature space F has its own distribution, which is represented by the number of different statistical characteristics over different smart devices. The analysis of such distributions can be useful in order to identify which features are most important for the classification. In this work, we considered three statistical characteristics of the distribution of each feature, such as: mean, median and standard deviation. Table 5 summarizes the statistical characteristics of each feature for both datasets. However, for illustration purposes we plot the probability distribution of the features under consideration for the first dataset only, as shown in Fig. 2. As can be seen, the interarrival time shows a Gaussian distribution (as explained in the previous subsection), while all other features illustrate an exponential distribution.

TABLE 5. Statistical characteristics of IoT traffic features.

Datasets	Measures	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
Dataset 1	Mean	0.45	0.67	0.59	0.50	0.35	0.38	0.37	0.07	0.16	0.74	0.69
	Median	0.50	0.85	0.75	0.65	0.06	0.13	0.24	0.02	0.03	0.86	0.81
	S.D	0.22	0.28	0.25	0.45	0.37	0.37	0.26	0.18	0.30	0.28	0.28
Dataset 2	Mean	0.39	0.001	5.21	0.29	0.38	0.44	0.33	0.09	0.01	0.61	0.46
	Median	0.38	1.45	1.43	0.33	0.11	0.67	0.25	0.01	0.01	0.69	0.39
	S.D	0.39	0.03	0.002	0.12	0.36	0.34	0.20	0.22	0.01	0.19	0.18

C. FEATURE CORRELATION

One very important aspect of the performance of the classification is the correlation between the features. Hence, in this work we consider the feature correlation from two perspectives. Firstly, we examine which features are correlated within the feature space. The correlation between two features say, f_i and f_j, is calculated using the Pearson's correlation coefficient which is given as:

$$\rho_{(f_i,f_j)} = \frac{cov(f_i, f_j)}{\sigma_{f_i}\sigma_{f_j}} \tag{1}$$

where $cov(f_i, f_j)$ is the covariance between features f_i and f_j, whereas $\sigma_{(f_i)}$ and $\sigma_{(f_j)}$ represent the standard deviation of the i^{th} and j^{th} feature respectively. The value of correlation coefficient lies between -1 and 1. If there is no correlation between the features f_i and f_j then $\rho_{(f_i,f_j)} = 0$. A perfect negative correlation is found if $\rho_{(f_i,f_j)} = -1$ and a perfect positive correlation is found if $\rho_{(f_i,f_j)} = 1$. We plot the correlation between features for the first dataset as a heatmap, which is shown in Fig. 3.

As it can be seen, the source IP address is more correlated to TTL, destination port number, source MAC addresses and destination IP addresses. Furthermore, the destination IP address and source port number, the destination IP address and destination MAC address, the packet length and destination MAC address, the source MAC address and source port number, the source port number and destination port number are also highly correlated features.

Secondly, we find the correlation between the input vector features and the target class labels. Then based on the relationship between independent variables (i.e., feature space) and dependent variable (i.e., class label) we select the features for our learning (classification) framework. This is further discussed in Section IV-C.

IV. PROPOSED CLASSIFICATION FRAMEWORK

A. OVERVIEW

The proposed classification framework consists of three key steps as shown in Fig. 4 and discussed in the following sections.

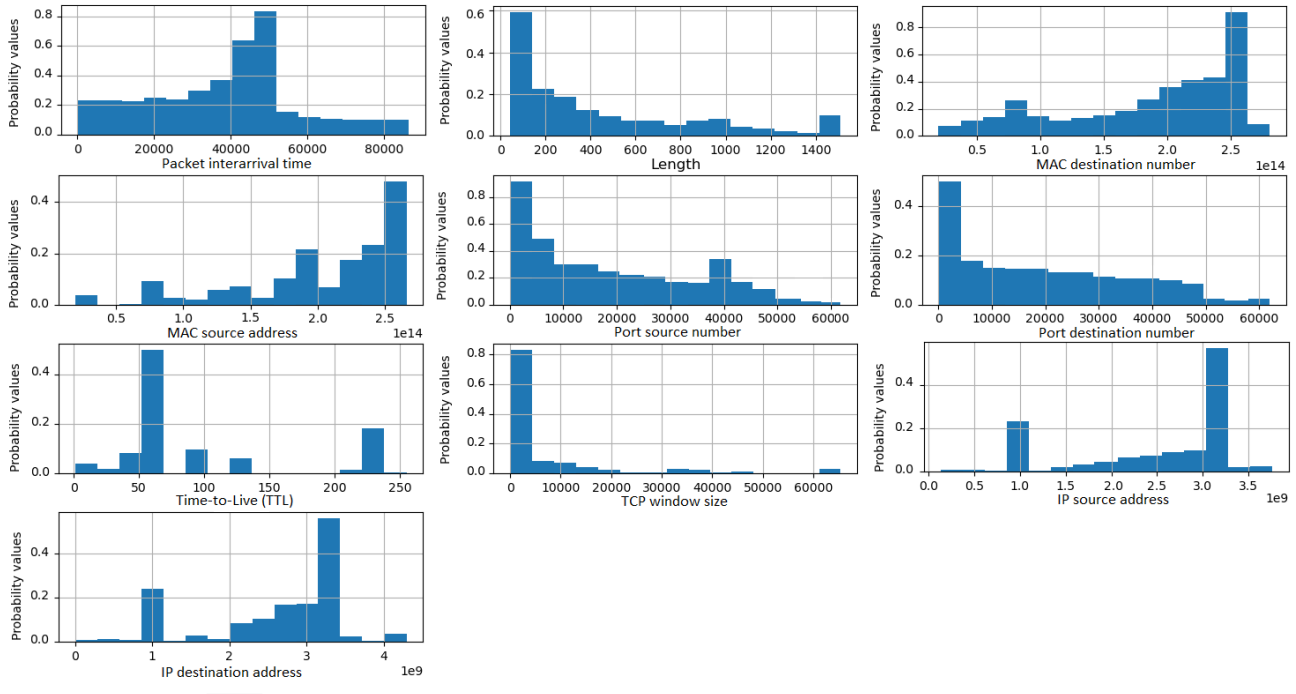


FIGURE 2. Probability distributions of IoT traffic flow features of Dataset 1.

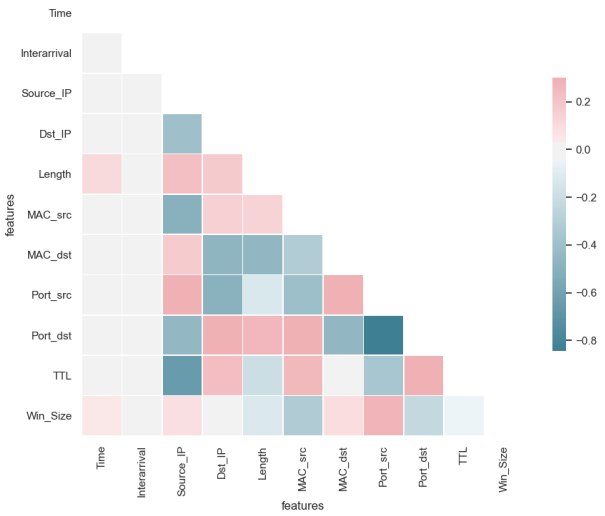


FIGURE 3. Correlation between IoT traffic features of Dataset 1.

1) **Preprocessing the IoT Traffic (Section IV-B):** It is the first step executed and it aims at providing the weighted preprocessing of dataset along with the rescaling, imputation and transformation of traffic traces.
2) **Selecting the most relevant features (Section IV-C):** It consists of the selection of the most important features, which are highly correlated to the class labels, using the ANOVA filter based selection method.
3) **Two-stage learning model (Section IV-D):** Here the classification of the IoT traffic traces is done using the proposed two stage learning model. At stage 0, the classification is performed by applying a logistic regression technique, while the tentative classes are provided. At stage 1, a neural network is applied to provide the final classes.

The operational flow of the proposed work is provided in Fig. 5.

B. DATA PREPROCESSING

During the data preprocessing, a basic filtering of the dataset is performed in order to remove some of the non-meaningful packets such as ping, DNS requests, etc. The features such as TTL, window size, packet length are already numerical, whereas the interarrival time feature is converted to seconds. Following, we observed that some of the features such as "set of port numbers (f_5 and f_6)", "set of IP addresses (f_2 and f_3)" and "set of MAC addresses (f_{10} and f_{11})" are nominal and multi-valued (having more than one value with a single data instance). As machine learning classifiers cannot deal with such data, we converted these features into a numerical form using a two-step procedure.

Firstly, we perform the data cleaning by passing the nominal vectors to the Bag-of-Word (BoW) model [26]. Secondly, as the BoW assigns the same importance to each vector word, we have proposed a relevance weighting to assign a prioritized importance to each word within each vector. These relevance weights, attributed to each feature vector, are passed to the stage 0 classifier and is given by Eq. (2):

$$Relevance\ Weight = wf_{w,v} \times vf_{w,v} \tag{2}$$

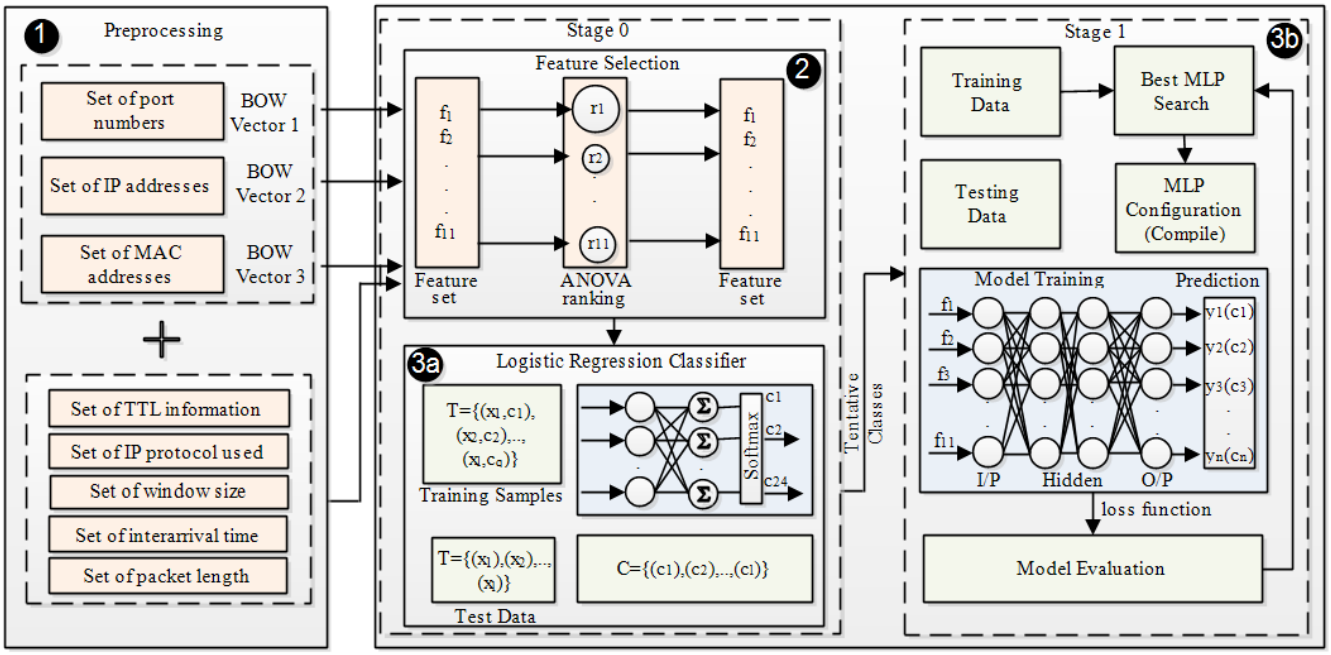


FIGURE 4. Overview of proposed two-stage classification framework.

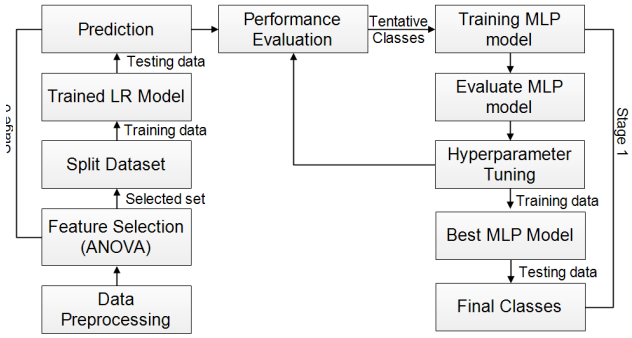


FIGURE 5. Operational flow of the proposed work.

where $wf_{w,v}$ denotes the word frequency of a word w within a vector v and $vf_{w,v}$ represents the total vector frequency. Herein, the vectors consist of the "port numbers vector", "IP addresses vector", and "MAC addresses vector". The word frequency $wf_{w,v}$ is defined as the number of times that w occurs in v and is given using Eq. (3):

$$wf_{w,v} = \frac{number\ of\ \ occurrence\ of\ \ a\ word\ in\ \ a\ vector}{number\ of\ \ words\ in\ \ that\ vector} \quad (3)$$

Because frequent words are less informative than rare words, the vector frequency, $vf_{w,v}$ is given as Eq. (4).

$$vf_{w,v} = \log \frac{number\ of\ \ vectors}{number\ of\ \ vectors\ containing\ word\ w} \quad (4)$$

After this step, we impute the missing values of features using their mean value and re-scale the dataset between 0 and 1 using the MinMaxScaler technique.

C. FEATURE SELECTION

The supervised feature selection is a way to choose the input features that are believed to be the most useful to a model in order to predict the target variable. For our supervised feature selection method, we resort to either wrapper methods or filter based methods. A wrapper based method, such as Recursive Feature Elimination (RFE), selects the features that are performing well.

However, for the selection of features from our feature space F, we employed the filter-based feature selection technique [27] which uses the statistical methods to score the relationship between the features and the target labels i.e., class labels. Specifically, we have selected the ANOVA (Analysis of Variance) F-value feature selection technique because our input features are quantitative or become quantitative after preprocessing and the target class labels are of categorical nature (i.e. c_1 indicates a belkin wemo switch, c_2 represents smart cam and so on).

D. PROPOSED TWO-STAGE LEARNING MODEL

1) STAGE 0 CLASSIFIER

The Logistic Regression method is employed at stage 0, which takes the selected set of features for the training, as given by the ANOVA F-value. The reason that we have selected this classifier is that it has been proven to perform well for very large data sets [28], as in the case of a smart environment. The logistic regression technique investigates the association among the independent variables and the dependent variables of the problem. In our scenario, the selected features are the independent variables and the device categories (e.g. hubs, cameras, etc.) are the dependent variables.

The goal is to estimate the probability p for a combination of independent variables using the following logit function:

$$logit(p) = ln\frac{p}{1-p} \tag{5}$$

where ln is the natural logarithm and p denotes the probability of an independent variable. The anti log of (5) allows us to find the estimated regression equation given by Eq. (6):

$$\begin{aligned} logit(p) &= ln\frac{p}{1-p} \\ &= \beta_0 + \beta_1 * x_1 + \beta_2 * x_2 + \ldots + \beta_n * x_n \Rightarrow \\ p &= \frac{e^{\beta_0+\beta_1*x_1+\beta_2*x_2+\ldots+\beta_n*x_n}}{1+e^{\beta_0+\beta_1*x_1+\beta_2*x_2+\ldots+\beta_n*x_n}} \end{aligned} \tag{6}$$

where β_0 is an intercept, β_1, β_2, and β_n are the regression coefficients, x_1 is the first independent variable, x_2 is the second independent variable, and x_n is the n^{th} selected feature. In order to calculate β coefficients, we employed the Gradient Descent method [29]. The general form of Eq. (6) is given as:

$$p(y_i|x_1, x_2, \ldots, x_n) = \frac{1}{1+e^{-(\beta_0+\beta_1*x_1+\beta_2*x_2+\ldots+\beta_n*x_n)}} \tag{7}$$

where y_i represents the dependent variable i.e., the i^{th} IoT device class, which we predict based on x_1, x_2, and x_n. After calculating the regression coefficients the testing component comes into effect, where the classifier uses the regression coefficients and computes the estimated regression for each testing instance using Eq. (7). Finally, stage 0 classifier performs a first tentative prediction.

2) STAGE 1 CLASSIFIER

In order to optimally classify the IoT devices, we architect the Multi-Layer Perceptron Artificial Neural Network (MLP-ANN) [30] based classification as our stage 1 classifier. MLP-ANNs are composed of multiple neurons that are arranged in the form of an input, output, and hidden layers. In this work, the architecture of MLP-ANN consists of one input layer with 11 neurons, because we have 11 different features to be passed as an input to the neural network. Following, we optimize the number of hidden layers, while the output layer consists of n number of neurons depending on the number of labelled classes n found in each of the dataset.

MLP-ANN provides two major processes for the classification task. Firstly, it performs the forward propagation process, which feeds the features to the input layer neurons. In our case, all quantitative features along with the output from stage 0 classifier (i.e., tentative classes) are fed to an input layer. Following, the input layer propagates these data to the hidden layers and then to the output layer. The neurons in each of the neural network layer calculates the weighted sum as output which is then passed to the activation function and is given by Eq. (8).

$$O_i^{(l)} = V^{(l)}(\sum_j w_{(i,j)}^{(l)} \times O_j^{(l-1)} + B_i^{(l)}) \tag{8}$$

where the superscripts on variables represent the layer number and the subscripts represent the neuron numbers in the respective layer. The $w_{(i,j)}^{(l)}$ denotes the weight of a connection between the i^{th} neuron of layer l and the j^{th} neuron of layer $l-1$; $B_i^{(l)}$ represents the bias value applied at the l^{th} layer for the i^{th} neuron; $O_i^{(l)}$ denotes the output of the the i^{th} neuron at the l^{th} layer and V^l represents the nonlinear activation function applied at layer l. This work applied the Rectified Linear Units (ReLU) activation function at the input layer and the softmax activation function at the output layer.

The above process continues till the output layer predicts a label, i.e., class of an IoT device, which is then compared with the actual label and a loss value is calculated using a loss function based on the categorical cross entropy. Secondly, a back propagation is done in which weights are updated using the predicted output, desired output and their difference. The goal is to minimize the loss by finding the optimal weights value. The optimization function that we applied is based on the Adaptive Moment Estimation (Adam) because it is proved to be very robust for large datasets [31].

To model an optimal MLP-ANN, we used the Keras tuner [32] along with the Random Search technique. For the hyper parameter optimization, we determine the optimal number of hidden layers, the optimal number of neurons in each layer (i.e., a search between 22 and 512 neurons), and the learning rate (i.e., a search between 1e-2 and 1e-4) using a random search tuner. Following, these parameters are passed to the Adam optimizer, since we want to achieve the best performance along with the least computational complexity.

V. CLASSIFICATION ALGORITHM

A. ALGORITHM DESCRIPTION

The preprocessing algorithm (Algorithm 1) consists of the *PREP* procedure, which firstly generates the BoW representations using the function *generate_BOW*(). Then, the relevant weights are calculated by employing the *word_Freq*() and *vector_Freq*() functions, which takes BoW as an input. Following, the features are scaled using the function *MinMaxScaler*(). Algorithm 2 depicts the learning model consisting of two procedures, namely, *LOGREG* and *MLP*. In the *LOGREG* procedure, the input labels x and output labels y are split into training and testing data using the function, *split*(). Next, the filter-based feature selection is done using the statistical method called ANOVA score and this is achieved by employing the *SelectKBest*() function. Then the *LogisticRegression*() generates and fit the model using the *fit*() function. The prediction is done using the *predict*() which contains the *x_tst* as testing dataset.

The *MLP* procedure generates the classification results based on the MLP-ANN which takes stage's 0 results along-with the other features. At this stage, firstly the data are split using *split*() and then a sequential model is created using the function, *build_model*(). Following, the keras tuner is applied to search the number of models using *RandomSearch*(), which takes the sequential model, the number of trials per search, the max trials allowed and the search objective as an input. Then, the *getBestModel*() returns the

Algorithm 1 Preprocessing Algorithm

PREP(f_2,f_3,f_5,f_6 ,f_{10},f_{11},**devices**)
// f_2 and f_3 are source and destination IP addresses; f_5 and f_6 are source and destination port numbers; f_{10} and f_{11} are source and destination MAC addresses; and *devices* labels.
1. $BOW_1 \leftarrow generate\_BOW(f_2, f_3)$
2. $BOW_2 \leftarrow generate\_BOW(f_5, f_6)$
3. $BOW_3 \leftarrow generate\_BOW(f_{10}, f_{11})$
4. $wf \leftarrow word\_Freq(BOW_1, BOW_2, BOW_3)$
5. $vf \leftarrow vector\_Freq(BOW_1, BOW_2, BOW_3)$
6. $rel_{weight} \leftarrow wf \times vf$
7. set $x \leftarrow dataset(BOW_1, BOW_2, BOW_3, rel_{weight})$
8. set $y \leftarrow dataset(devices)$
9. set $x_{norm} \leftarrow MinMaxScaler(x)$
Output: $\mathbf{x}_{norm}$,y

model with the highest validation accuracy across all models given by the *RandomSearch*(). Finally, we fit the model with *fit*() for 70 epochs and then call the *predict*() function.

Algorithm 2 Learning Algorithm

LOGREG($\mathbf{x}_{norm}$,y)
// x_{norm} is the dataset instances and y is the class labels
1. set $x_{tr}, x_{tst}, y_{tr}, y_{tst} \leftarrow split(x, y, test_{size} \leftarrow 0.2)$
2. set $x_{tr} \leftarrow selectKBest(Anova_{score}, x_{tr})$
3. set $x_{tst} \leftarrow selectKBest(Anova_{score}, x_{tst})$
4. set $model \leftarrow LogisticRegression(max_{iter} \leftarrow 3000)$
5. set $fit \leftarrow model.fit(x_{tr}, y_{tr})$
6. set $y_{pred} \leftarrow model.predict(x_{tst})$
Output: y_{pred} ▷ Stage 0
MLP($\mathbf{y}_{pred}$,$\mathbf{f}_1$,$\mathbf{f}_4$,$\mathbf{f}_7$,$\mathbf{f}_8$,$\mathbf{f}_9$, **devices**)
// y_{pred} is the output of Stage 0 classifier; f_1 is the interarrival time; f_4 is the IP protocol used; f_7 is the TTL; f_8 and f_9 are the window size and packet length; *devices* are the class labels.
7. set $x \leftarrow dataset(y_{pred}, f_1, f_4, f_7, f_8, f_9)$
8. set $y \leftarrow dataset(devices)$
9. set $x_{tr}, x_{tst}, y_{tr}, y_{tst} \leftarrow split(x, y, test_{size} \leftarrow 0.2)$
10. set $m \leftarrow build\_model()$
11. set $tuner \leftarrow RandomSearch(m, tuner.obj(val_{acc}), max_{tr} \leftarrow 3, search_{tr} \leftarrow 1)$
12. set $model \leftarrow tuner.getBestModel(num_{models} \leftarrow 1)$
13. set $history \leftarrow model.fit(x_{tr}, y_{tr}, epochs \leftarrow 70)$
14. set $y_{pred} \leftarrow model.predict(x_{tst})$
Output: y_{pred} : $FS \leftarrow devices$ ▷ Stage 1

B. ASYMPTOTIC ANALYSIS

Proposition 1: The computational complexity of PREP procedure is O(n)

Proof: The PREP procedure running time depends on the number of feature vectors, represented as n. Lines 1-3 take a constant time as they split the vectors into words, thus $O(1)$. Lines 4-5 and 7-8 are assignment statements and each requires $O(1)$ operations. For the rel_{weight} statement (line 6) the complexity is $O(1) * O(n) = O(n)$. However, line 9 depends on the number of feature vectors n and thus, in the worst-case scenario needs $O(n)$. Accordingly, the overall time complexity of PREP procedure is linear i.e., $O(1) + O(1) + O(n) + O(n) = O(n)$. □

Proposition 2: The computational complexity of LOGREG procedure is O(n).

Proof: Line 1 is a simple assignment statement (i.e., $O(1)$) and lines 2-3 require $O(n)$ computation time in the worst scenario. Regarding the training time (lines 4-5) of LOGREG the complexity is $O(t * n)$ where t is the number of training examples and n is the number of selected data features used for the classifier training. Additionally, the testing time taken by line 6 is $O(n)$. Thus, the LOGREG takes $O(1)+O(n)+O(t*n)+O(n) = O(n)$, which can be beneficial for low latency applications that require a fast classification method. □

Proposition 3: The computational complexity of MLP procedure is O(nd)

Proof: In the *MLP* procedure, lines 7-9 consist of simple assignments i.e., $O(1)$. Line 10 indicates the *build_model*() function of the neural network and its complexity is $O(n * d * t * e)$, where for proposition 3, n represents the number of layers, d denotes the number of neurons in each layer, t is the number of training examples and e is the number of epochs. Because we are using 80% training examples i.e., 664796 for 70 epochs, the complexity for this part is $O(n*d*664796*70) = O(nd)$. Following, *RandomSearch*() (line 11) takes $O(n)$ for the worst scenario and line 12 takes a constant amount of time i.e., $O(1)$. Line 13 takes $O(t)$ and testing time taken by the line 14 is $O(n)$. Thus, the *MLP* takes $O(1) + O(nd) + O(n) + O(1) + O(t) + O(n) = O(nd)$ time. □

The overall complexity, T of the proposed learning framework is represented in term of n as: $T(n) = O(n) + O(n) + O(nd) = O(n)$. Thus, it is a linear time learning work.

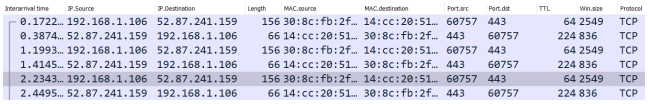
VI. PERFORMANCE EVALUATION

A. MODEL IMPLEMENTATION AND FRAMEWORKS

1) DATASET DESCRIPTION

In this work, we have used two different datasets provided by [33] and [25] consisting of IoT traffic traces in a smart environment. The description of both datasets is provided as follows:

Dataset 1 [33] consists of network traffic traces from 28 smart devices. As we have considered a subset of the network traffic, which is a total of 12000317 labeled instances of 22 IoT devices, for this dataset we have 22 distinctive classes. The devices are namely, smart phone, belkin wemo switch, belkin wemo motion sensor, dropcam, HP printer, iphone, laptop, nest protect smoke alarm, netatmo welcome, netatmo weather station, PIX star photo frame, samsung tab, samsung smartcam, smart things, TP link camera, TP link



Interarrival time	IP.Source	IP.Destination	Length	MAC.source	MAC.destination	Port.src	Port.dst	TTL	Win.size	Protocol
0.1722…	192.168.1.106	52.87.241.159	156	30:8c:fb:2f…	14:cc:20:51…	60757	443	64	2549	TCP
0.3874…	52.87.241.159	192.168.1.106	66	14:cc:20:51…	30:8c:fb:2f…	443	60757	224	836	TCP
1.1993…	192.168.1.106	52.87.241.159	156	30:8c:fb:2f…	14:cc:20:51…	60757	443	64	2549	TCP
1.4145…	52.87.241.159	192.168.1.106	66	14:cc:20:51…	30:8c:fb:2f…	443	60757	224	836	TCP
2.2343…	192.168.1.106	52.87.241.159	156	30:8c:fb:2f…	14:cc:20:51…	60757	443	64	2549	TCP
2.4495…	52.87.241.159	192.168.1.106	66	14:cc:20:51…	30:8c:fb:2f…	443	60757	224	836	TCP

FIGURE 6. Samples of IoT traffic traces from dataset 1.

plug, TP link router, triby speaker, withings smart baby monitor, withings smart scale, ipv4mcast and amazon echo.

Dataset 2 [25] consists of traffic traces of from 81 IoT devices which are located at various US and UK locations. These devices belongs to cameras, smart hubs, home automation, TVs, audio devices and home appliances categories. For the second dataset, a total of 40588450 labeled instances of 68 IoT devices were used in this work.

A sample of the network trace used from the first dataset is provided in Fig. 6. Nonetheless, since we have used the same feature space for both datasets, Fig. 6 reflects the traces from the second dataset as well. The feature called ''MAC address'' of each device is used to provide the label to each network trace in both of the datasets.

2) EXPERIMENT SETUP

The configuration settings used for our experiments and for both datasets are listed in Table 6. The proposed model was implemented in Python (version 3.8.2). In Table 6, the No. of architectures represents the number of different classification solutions used during our experimentation. These architectures/solutions are further explained in section VI.3. Following, the total number of instances provides the number of labelled instances used from each dataset and the total number of classes represents the total number of distinct device types. The reason that we have selected a subset of the labelled instances for each dataset, is because these datasets span over a period of about two months and the training of such a large amount of data can create several big data challenges. Furthermore, as shown later, we also managed to achieve a very good performance by using only the specific subset of these datasets. Accordingly, the selected subset of data under evaluation resulted in a slightly reduced number of classes for each dataset.

Regarding the number of tuner trials, this value represents the keras tuner trials that we executed for our proposed model. In more details, for the first dataset, we noticed that after 5 trials we have achieved the best hyperparameter configuration and for the second dataset after 3 trials. The reason for executing several trials, is that the keras tuner uses a different set of parameters (i.e. learning rate, number of layers and number of neurons in each layer) at each trial and then it selects the best performing configuration. Nonetheless, we have not seen a significant variation between the accuracy of the different trials. Lastly, we split both of the dataset instances into three groups as: 60% training instances, 20% validation and 20% testing instances, which is a common split ratio in the machine learning domain.

For the evaluation of the classification performance, we have considered the following well known classification metrics:

1) Precision: It is the ability of a classifier to not label an instance that is actually negative as positive and is given as:

$$Precision = \frac{TruePositive}{TruePositive + FalsePositive} \tag{9}$$

2) Recall: Recall calculates the rate of all the positive instances, which is also called true positive rate and is given as:

$$Recall = \frac{TruePositive}{TruePositive + FalseNegative} \tag{10}$$

3) F1-score: It is the harmonic mean of the precision and recall metrics and is given as:

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{11}$$

4) Accuracy: It is the proportion of correctly classified instances and is given as:

$$Accuracy = \frac{CorrectPredictions}{TotalPredictions} \tag{12}$$

5) Confusion matrix: It is a table that is used to describe the classifier performance on a set of test data for which the true values are known.

The values of recall, precision, F1-score, confusion matrix and accuracy are calculated between [0,1] with 1 indicating the best and 0 the worst performance. However, a decrease from 1 towards 0 is good for the loss function of the network.

3) ARCHITECTURE MODELS

We have applied different composite models consisting of neural networks along with traditional machine learning algorithms to see their suitability for the IoT traffic multi classification problem. Table 7 provides the description of the different network architectures. The LR represents the logistic regression algorithm and GB denotes the gradient boosting algorithm (architecture I) [24]. The NB is Naive Bayes algorithm at stage 0 and RF denotes applying random forest at stage 1 (architecture II) [23]. IP(x) stands for the input layer of neural network with x number of neurons. FC(x) denotes the fully connected layer of neural network with x number of nodes (or neurons). OP(x) represents the output layer of neural network with x number of classes i.e., neurons.

MLP represents the multi layer Perceptron neural network with an input layer consisting of 11 neurons, two fully connected layer and one output layer with 22 classes (architecture III). LR(RFE)+MLP denotes the logistic regression at stage 0 with recursive feature elimination method and MLP at stage 1 with one input layer, two fully connected layers and one output layer (architecture IV). LR(Anova)+MLP (keras tuner) denotes the logistic regression at stage 0 with

TABLE 6. Configurations used in the experiments.

N/W Settings	No. of Architectures	Total Instances	Total Classes	No. of Tuner Trials	Data Split	Metrics
Dataset 1	5	12000317	24	5	60:20:20	Precision/Recall/F1/Accuracy
Dataset 2	5	40588450	68	3	60:20:20	Precision/Recall/F1/Accuracy

TABLE 7. Description of model architectures applied to the multi-classification problem.

Architecture	Details
I: LR+GB	Stage 0: LR - Stage 1: GB
II: NB+RF	Stage 0: NB - Stage 1: RF
III: MLP	Single Stage : MLP with IP(11)-FC (100)-FC(100)-OP(22)
IV:. LR(RFE)+MLP	Stage 0: LR along with RFE - Stage 1: MLP with IP(11)-FC(100)-FC(100)-OP(22)
V: LR(Anova)+MLP(Keras Tuner)	Stage 0: LR along with Anova-score feature selection - Stage 1: MLP with IP(11)-FC (n)-FC (n)-OP(22)

the Anova based feature selection and MLP at stage 1 (architecture V), which is the two-stage learning model proposed in this paper.

For comparison purposes, it is important to mention that the accuracy of existing works are less than the proposed framework, as shown in the following subsection. For example, the proposed framework in [16] achieves an accuracy of 99.0%, the authors in [21] achieve 96% accuracy, while in [22] the accuracy is 99.2%. However, for our evaluation, we compared the proposed framework with the architecture I [24] and architecture II [23], which both use the first dataset.

Additionally, to better illustrate the efficiency of our work, we also compare our proposed architecture V with the architectures III and IV which are based on the MLP neural network. For all the neural network-based architectures (i.e. III to V), the training was done with a number of epochs between 50 and 100. The training was stopped earlier if an increase in the number of epochs did not lead into an improvement of the loss function.

Furthermore, for the activation functions we used the ReLU along with the softmax activation which was applied at the last output layer. The loss functions used was the categorical cross entropy. Finally, the optimization was done with the Adaptive Gradient (AdaGrad) for the architectures III and IV and with Adam for architecture V. The particular configurations gave the best results for each of the examined architectures.

We have also experimented with different LSTM configurations. In particular, we executed five tuner trials to find the best hyperparameters such as number of layers, LSTM units, learning rate, etc. However, these models gave less accurate results, (i.e., 70% of accuracy). Moreover, we also considered the AdaGrad optimizer for the architecture V but it produced an accuracy of 85% and we decided to show only the results of the best configuration, which uses the Adam optimizer.

B. RESULTS

1) IMPACT OF ARCHITECTURES

a: STAGE 0

Fig. 7 illustrates the performance of the different network architectures at stage 0, in terms of precision, recall and F1 score for both datasets. We have only considered the architectures I, II, IV, and V for this part, because architecture III i.e., MLP does not consist of two stages. In terms of the precision, our proposed architecture V provides the highest value i.e., 0.74 followed by LR(RFE) + MLP with 0.72 and LR+GB with 0.69 value for the first dataset. Regarding the second dataset, the same trend is noticed, as architecture V provides the highest value i.e., 0.87 followed by LR(RFE) + MLP with 0.83 and LR+GB with a value of 0.79.

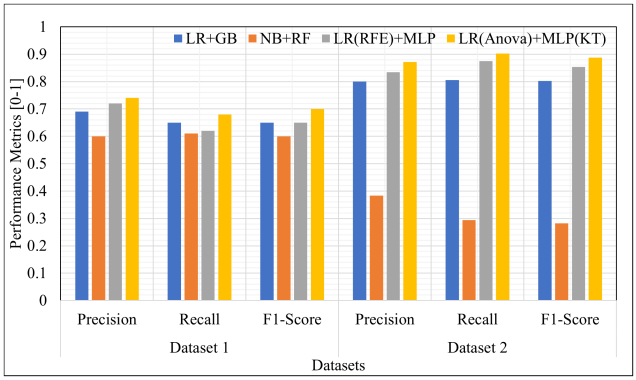


FIGURE 7. Performance comparison at stage 0.

In contrast, NB + RF performed poorly for both datasets, i.e., 0.6 for the first dataset and 0.4 for the second. This means that 40% of the labelled instances were wrongly classified as positive for the first dataset and 60% were wrongly classified as positive for the second. This can be attributed to the fact that the precision values of some devices were zero and less than 0.17 for many other. As an example, in the first dataset the most misclassified devices for the NB+RF were the Belkin Switch, HP printer, Netatmo Welcome, PIX-STAR, Samsung tab and TP link camera.

When looking into the recall metric, we see that the proposed architecture V also outperformed the rest of the models, followed by the LR+GB and LR(RFE)+MLP for the first dataset. However, for the second dataset, LR(RFE)+MLP(KT) is followed by LR(RFE)+MLP and LR+GB, while architecture V remains the most efficient solution. Once again NB+RF gives the least average recall for both datasets, with 0.61 and 0.29 for dataset 1 and 2. The reason for this behavior is that the majority of instances were

100% misclassified. For instance, for the first dataset, out of 22 classes, instances of 8 classes were 100% incorrectly classified.

Lastly, we observe that the architecture V gives the highest value of F1 score among all architectures at stage 0, with a value of 0.7 for the first dataset, followed by LR+GB and LR(RFE)+MLP which both give an F1-score of around 0.65, whereas NB+RF achieves only 0.6. For the second dataset, our proposed architecture presents a F1-score of 0.89 followed by LR(RFE)+MLP, LR+GB, and NB+RF which give a F1-score of 0.85, 0.80, and 0.28 respectively.

b: STAGE 1

At this stage all five network architectures are considered as shown in Fig. 8 for both datasets. Moreover, we also included the accuracy in our evaluation metrics, since the output of Stage 1 is our final classification. As it can be seen, our proposed architecture (LR(Anova)+MLP(KT)) attained an accuracy of 0.999, a precision of 0.996, a recall of 0.995 and a F1-score of 0.996 for the first dataset. Regarding, the second dataset, it achieved an accuracy of 0.998, a precision of 0.996, a recall of 0.997 and a F1-score of 0.997. Furthermore, LR(RFE)+MLP(KT) provided reasonable results followed by the other architectures for both of the datasets.

Once again, NB+RF continued to under-perform for both datasets at stage 1. Specifically, for the dataset 1, the NB+RF achieved a performance of only 0.78 for recall, 0.8 for precision and 0.77 for F1-score because 3335 training instances of Belkin switch class, 374 instances of HP printer class, 262 instances of the TP link camera class and 31 iPhone class instances were incorrectly classified. Similarly, for the dataset 2, the particular model achieved a performance of only 0.33 for recall, 0.29 for precision and 0.31 for F1-score because many instances of devices such as Tphilips Hub US, TP link bulb US, Sousvide US, TP link plug UK, T wemo plug UK, T wemo plug US, Wans view cam wired US, wans view cam wired UK, smart thing hub UK,sousvide UK,T philips hub UK,TP link bulb UK,TP link plug US were incorrectly classified.

Additionally, the NB+RF provided an accuracy of 0.77 for dataset 1 and 0.92 for dataset 2. Further analysis showed that for the first dataset, there were 5 classes incorrectly classified out of 22 and for the second dataset, there were 13 misclassified classes out of 68. As accuracy is the ratio of these numbers, we corroborate the poor performance of architecture II as shown in Fig. 8.

After analyzing the results of stage 1, we conclude that our architecture V and its variation (architecture VI) provide the best classification results in terms of all performance metrics for both of the datasets. This is a significant observation that proves the robustness of our framework that works equally well for different datasets with different number of classes. That is not the case for architectures I-III, which presented a great deviation in the attained results between the two datasets.

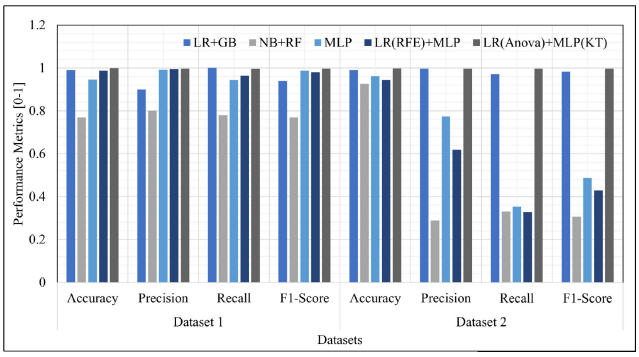


FIGURE 8. Performance comparison at stage 1.

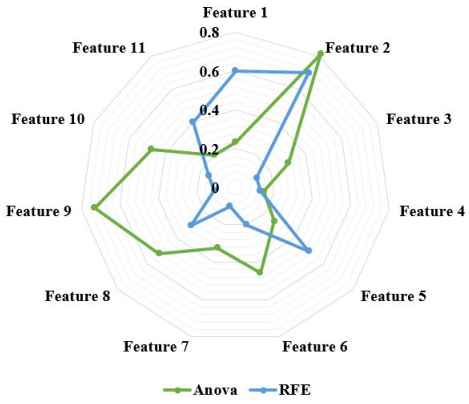


FIGURE 9. Feature ranks provided by the feature selection methods.

2) IMPACT OF FEATURES

Fig. 3 illustrated the correlation of the full set of features for the first dataset. However, it is critical to understand which features have a higher importance (rank value) provided by the feature selection method in the classification process. For this purpose, we provide the full set of features along with their ranks, as calculated by Anova score and RFE for dataset 1, in Fig. 9. The most important features selected for both datasets are provided in Table 8.

For the architectures I, II, III, we have used all features during the training and testing phases, thus, we only compare the architectures IV and V to see the feature importance. Specifically, we illustrate the ranks provided by the RFE for architecture IV and the ranks provided by the Anova score for architecture V. The rank values are between 0 and 1. It can be seen that the highest rank provided by Anova was 0.8 given to the feature 2 i.e., source IP address and the least rank given by Anova score was 0.14 for feature 4 i.e., IP protocol used by device. For the RFE method, the highest rank was provided to feature 2 i.e., 0.7 and the least to the feature 7 i.e., TTL information. The features were selected in decreasing order of their ranks by the architectures.

In more details, Table 8 provides the information about the features utilized by each architecture along with the performance metrics of each architecture for both datasets. The first three architectures used all 11 features. However,

TABLE 8. Classification performance metrics vs features employed.

Datasets	Architecture	Features	Precision	Recall	F1	Accuracy
Dataset 1	LR+GB	$f_1, f_2, f_3, f_4, f_5..., f_{11}$	0.900	1.000	0.940	0.990
	NB+RF	$f_1, f_2, f_3, f_4, f_5..., f_{11}$	0.800	0.780	0.770	0.770
	MLP	$f_1, f_2, f_3, f_4, f_5..., f_{11}$	0.992	0.943	0.986	0.946
	LR(RFE)+MLP	$f_2, f_1, f_5, f_{11}, f_8, f_6, f_{10}, f_4$	0.994	0.964	0.979	0.986
	LR(Anova)+MLP(Tuner)	$f_2, f_9, f_8, f_{10}, f_6, f_7, f_3, f_5$	0.996	0.995	0.996	0.999
Dataset 2	LR+GB	$f_1, f_2, f_3, f_4, f_5..., f_{11}$	0.997	0.972	0.984	0.990
	NB+RF	$f_1, f_2, f_3, f_4, f_5..., f_{11}$	0.289	0.331	0.307	0.926
	MLP	$f_1, f_2, f_3, f_4, f_5..., f_{11}$	0.774	0.354	0.486	0.961
	LR(RFE)+MLP	$f_5, f_6, f_8, f_{10}, f_{11}$	0.619	0.328	0.429	0.943
	LR(Anova)+MLP(Tuner)	f_4, f_5, f_6, f_7, f_8	0.997	0.997	0.997	0.999

as mentionned earlier, architecture IV selected the features by RFE and architecture V selected the features by ANOVA method. For the first dataset, the selected features by RFE for architecture IV consists of the source IP address (f_2), interarrival time (f_1), source port number (f_5), destination Ethernet address (f_{11}), window size (f_8), destination port number (f_6), source Ethernet address (f_{10}) and IP protocol used (f_4). In contrast, the selected features by Anova for architecture V consists of source IP address (f_2), packet length (f_9), window size (f_8), source Ethernet address (f_{10}), destination port number (f_6), TTL (f_7), destination IP address (f_3), and source port number (f_5).

For the second dataset, the selected features by RFE in architecture IV are the source port number (f_5), destination port number (f_6), window size (f_8), MAC address of source (f_{10}) and MAC address of destination (f_{11}). For the architecture V, the selected features are the type of protocol (f_4), port number of source (f_5), port number of destination (f_6), TTL (f_7) and window size (f_8). Therefore, source IP address, packet length, window size, source Ethernet address, destination port number, TTL, destination IP address, and source port number are more relevant to classify labels for dataset 1 and the features as protocol, port number of source, port number of destination,TTL and window size are more important for the classification in the second dataset.

To better illustrate the impact of feature selection in the resulted accuracy, we provide the following formal logic representation for the first dataset. Nonetheless, the same logic can be easily applied for the second dataset as well.

In more detail, we are representing the actual and selected feature sets of dataset 1 as: $R = \{f_2, f_1, f_5, f_{11}, f_8, f_6, f_{10}, f_4\}$ and $A = \{f_2, f_9, f_8, f_{10}, f_6, f_7, f_3, f_5\}$ respectively. According to these sets, we model $R \cap A$ as follows:

$$R \cap A = \{x | x \in R : x \in A\} \Leftrightarrow \{f_2, f_5, f_8, f_6, f_{10}\} \quad (13)$$

The intersection $R \cap A$ gives the features that were used by both architectures. However, in order to evaluate the impact of the feature selection in the overall performance, we need to identify the features that were not included in both architectures, which is captured as follows:

$$R - A = \{x | x \in R \wedge x \notin A\} \Leftrightarrow \{f_1, f_{11}, f_4\} \quad (14)$$

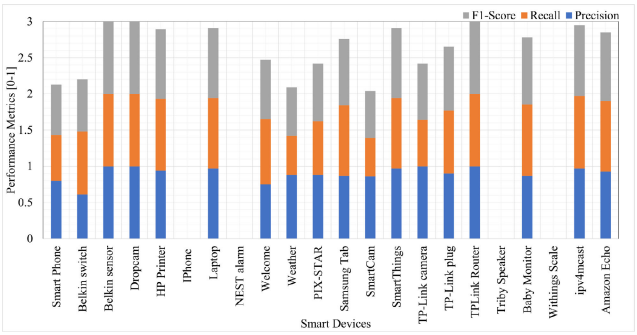


FIGURE 10. Performance comparison per device for architecture V.

Equation (14) provides the features that are only included by RFE and these are the interarrival time, the destination MAC address and the IP protocol used. Since, architecture IV presented an inferior performance than architecture V, we can safely say that these three features did not provide a well aligned information with the features given by $R \cap A$. Following, we extract the features included by the Anova score method but not from the RFE:

$$A - R = \{x | x \in A \wedge x \notin R\} \Leftrightarrow \{f_9, f_7, f_3\} \quad (15)$$

As (15) suggests, the packet length, TTL and destination IP address are the features that they are only considered by Anova and thus, by architecture V. Interestingly, we see that when these features are included in $R \cap A$ such that $(R \cap A) \cup (A - R) = A$, the performance increased significantly. Thus, the features $\{f_9, f_7, f_3\}$ have a positive impact in the performance of architecture V as they increased the accuracy to 99.9%, precision to 99.6%, recall to 99.5% and f1- score to 99.6% for dataset 1.

3) PERFORMANCE OF ARCHITECTURE V

In this part of the evaluation, we present the detailed results of the proposed architecture V for the first dataset, however, the accuracy, precision, recall and F1 score for both datasets can be found in Table 8, as shown earlier.

a: PERFORMANCE OF STAGE 0

As we have proved the superior performance of our proposed two-stage classifier (architecture V), in this part of the section

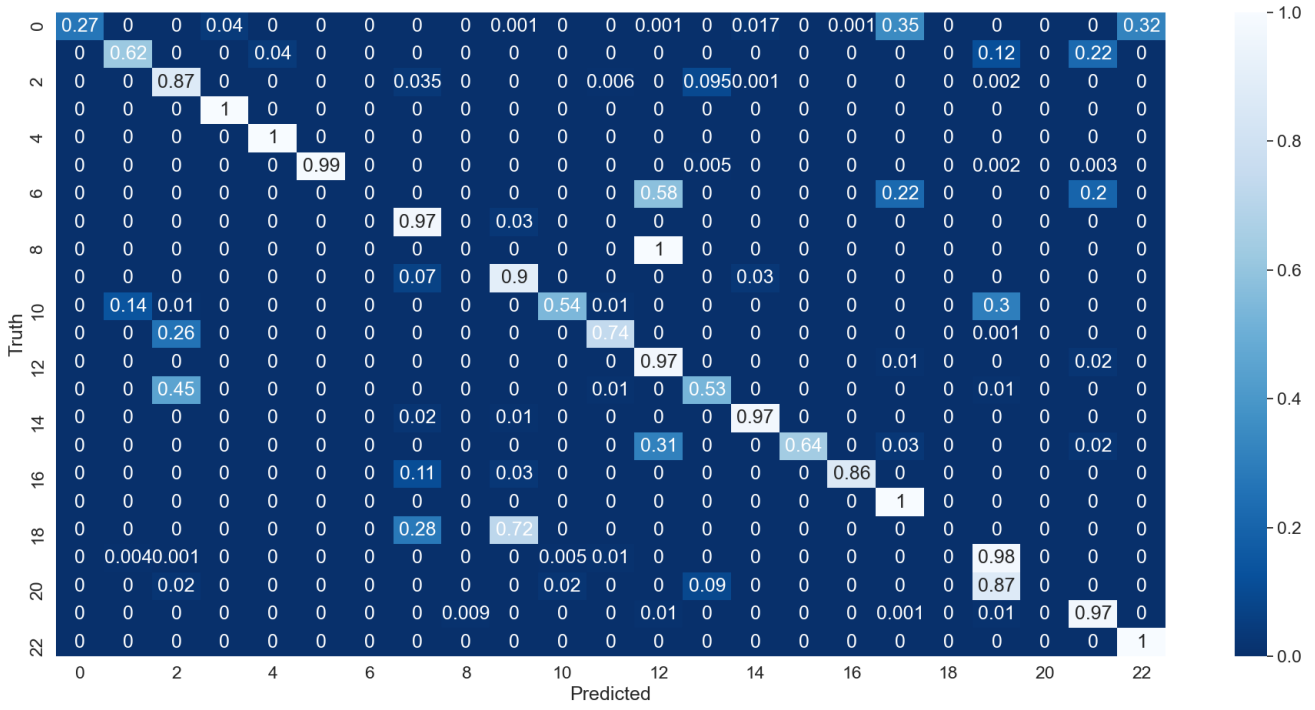


FIGURE 11. Confusion matrix for stage 0 of architecture V of dataset 1.

we delve into the details of the performance of the particular framework.

Accordingly, for the first dataset, Fig. 10 illustrates the performance metrics per device for stage 0. Some devices such as Belkin sensor, Dropcam and TP link router presents the highest performance, i.e., recall=1; precision=1 and F1-score=1, all aggregated to 3. The lowest precision is noticed for the belkin wemo switch i.e., 0.61, while the lowest recall and F1-score are observed for the Samsung smartcam i.e., 0.53 and 0.65 respectively. Furthermore, for the SmartCam the aggregated value is 2.04 since the F1 score is 0.65, the recall is 0.53, whereas the precision is significantly high, i.e., 0.86. For the Netatmo weather station device, the aggregated value is 2.09 as the precision is reasonably good, i.e., 0.88 but the recall and F1 score are relatively low i.e., 0.54 and 0.67. However, there were some devices such as withings scale, triby speaker, nest alarm, and iPhone for which precision, recall and F1-score were zero. The reason is that the instances of such devices were misclassified in other categories.

Following, we plot the confusion matrix of dataset 1 to give the overall performance of stage 0 as shown in Fig. 11. The row entries of a confusion matrix depict the actual values and the column entries depicts the predicted values for the 22 classes. All the diagonal entries correspond to correct classification whereas entries above diagonal are all Type I error (also called False Positive Rate (FPR)) and entries below are Type II error (also called False Negative Rate (FNR)). The goal is to minimize the Type I and Type II errors close or equal to zero.

At the main diagonal there are four exception cases: (i) the worst classification is noticed for the iPhone device, since 58% instances of the particular device were classified as Samsung galaxy tab, 22% instances were misclassified as TP link router, and 20% were misclassified as amazon echo thus depicting 100% FPR; (ii) for the nest protect smoke alarm the classification value is 0% with 100% FPR because it was misclassified as Samsung tab; (iii) for the triby speaker, we notice a 28% of misclassification as laptop (Type II error), and 72% of misclassification as netatmo welcome (Type II error); (iv) for the withings smart scale, we noticed 87% of misclassification as baby monitor (Type II error), 9.6% of misclassification as Samsung smartcam (Type II error), 1.9% of misclassification as Netatmo welcome, and 1.9% instances were incorrectly classified as belkin wemo switch.

This behavior is attributed to the following reasons: (a) there were 50 instances of iPhone compared to 3242, 87580 and 6231 of galaxy tab, TP link router and amazon echo instances; (b) 41 nest protect smoke alarm instances compared to 3242 instances of Samsung galaxy tab; (c) 771 triby speaker instances compared to 21815 laptop instances and 3995 instances of netatmo welcome; and (d) 52 withings smart scale instances compared to 5912, 4895, 3995 and 4407 instances of baby monitor, Samsung smart-cam, Netatmo welcome and belkin wemo switch respectively. Thus, the prediction value for these devices is much higher as compared to iPhone, nest protect smoke alarm, triby speaker and withings scale.

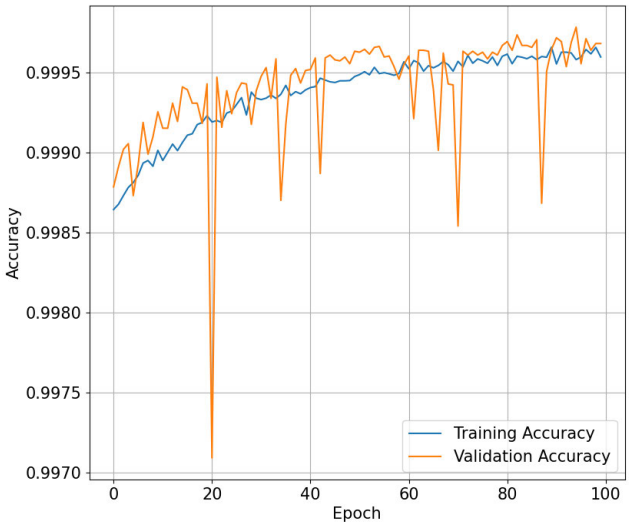


FIGURE 12. Training vs. validation accuracy of architecture V for 100 epochs.

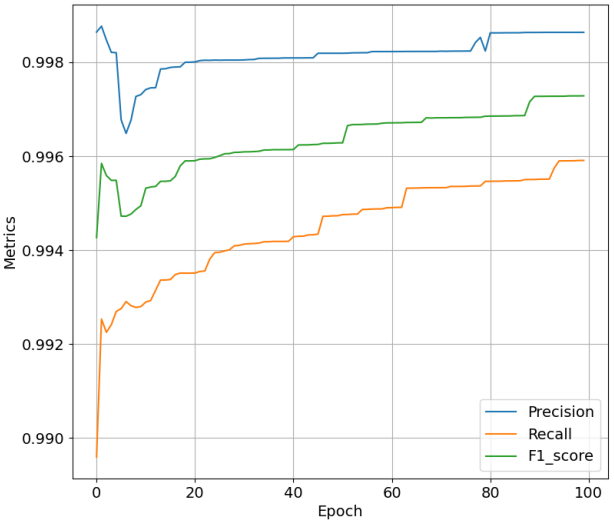


FIGURE 14. Comparison of performance metrics for stage 1 of architecture V over 100 epochs.

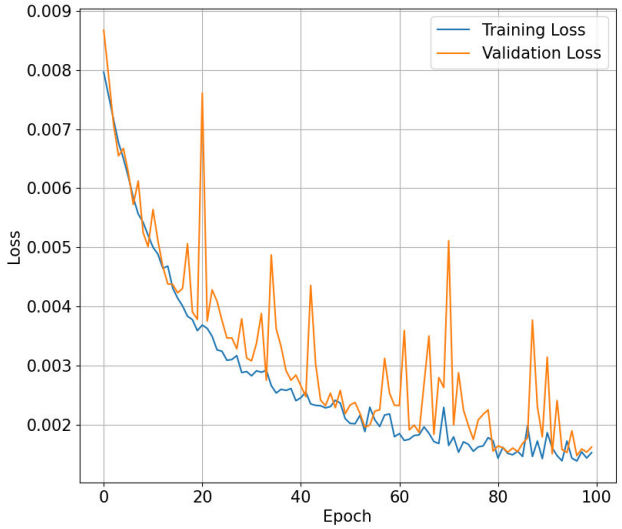


FIGURE 13. Training vs. Testing loss functions for stage 1 of architecture V.

b: PERFORMANCE OF STAGE 1

Fig. 12 depicts the training and testing accuracy, over the 100 epochs for the first dataset. The network model, i.e., optimized MLP at stage 1 of LR(Anova)+MLP (keras tuner), achieves better training accuracy i.e., 0.9997292 and validation accuracy i.e., 0.99962693 as the number of epochs increases. The initial accuracy values start from 0.998 at epoch 1 and the accuracy value does not change significantly after epoch 60. Regarding the spikes noticed, Keras Tuner estimates a close to optimal neural network topology using an exploitation versus exploration approach.

In the exploitation stage, it tries to improve the neural network topology, which output the most accurate results. In the exploration stage, it tries to randomly examine new neural network topologies that have not been explored yet. The exploration may help the optimisation process to escape from a local optimal, resulting however to the spikes noticed in Fig. 12. Yet, the optimisation process manage to converge due to this exploitation stage.

Following, we have plotted the loss function for the training and testing datasets across the 100 epochs as shown in Fig. 13. The learning curve shows the decay of the categorical cross entropy loss function with respect to the number of epochs. This curve is helpful in predicting whether our model is overfitted, underfitted or is fit to testing and training datasets. We see that the loss function for both training and testing decays to low values i.e., 0.001193 for training and 0.001516 for the testing datasets at epoch 100. The spikes are due to the use of a random search hyper tuner and the reasons discussed above. Furthermore, training and testing losses decrease and are stabilized around the same point i.e., after epoch 80 for training data. The model thus successfully captures the classification patterns.

Next, Fig. 14 depicts the performance metrics for 100 epochs at stage 1. The precision is high as compared to the other two performance metrics i.e., 0.996923 at the epoch 100. It can also be observed that the precision metric for the neural network does not exhibit significant changes after the epoch 80. Regarding the recall, it is lower compared to the precision and F1-score i.e., 0.9957 at epoch 100 and it shows a constant behavior after the epoch 95. For the F1-score, the value is 0.9964 at the epoch 90 and it does not present any significant changes after this point.

C. LIMITATIONS

Even though our framework provides very encouraging results, it still presents some limitations that stem from the intrinsic data nature of the IoT traffic multi-classification problem. This includes the extra overhead of monitoring the infrastructure to collect the traces, the construction of

a training dataset, and the computational overhead for the model training. In addition to that a classification task is a supervised learning approach. This means that if new types of IoT devices are connected in the local network a new cycle of data collection, annotation and training should begin in order to update the model.

VII. CONCLUSION

In this work, we studied the problem of IoT traffic classification. To solve this problem we have proposed a composite learning framework that consists of two stages. After an initial data preprocessing, the network traces are passed to stage 0, where a feature selection mechanism and a Logistic Regression classifier are applied. In particular, an ANOVA filter based selection technique decides on the most important features to be used by the stage 0 classifier. The tentative classification of the stage 0 classifier along with the remaining features were then passed to the stage 1 classifier, which used an optimal multi-layer perceptron neural network architecture that provides the final classification.

Following, a detailed experimentation and comparison with various composite architectures on two different IoT datasets have been performed. We concluded that the proposed framework can considerably increase the performance of the classification in terms of recall, precision, F1-score, accuracy and confusion matrix metrics. Regarding the accuracy, our proposed model achieved a 99.9% accuracy for the first dataset and a 99.8% accuracy for the second dataset, proving the generalization aspects of our approach.

The particular model is of utmost importance in an IoT to Cloud continuum communication model, where different IoT devices need to be classified and their traffic profiles be accurately predicted. This precise classification can positively contribute to the proper estimation of the required resources from the subsequent Edge and Cloud layers where the IoT traffic will be processed and analyzed.

The future direction of this work lies in the combination of our proposed model with a resource allocation mechanism that will be able to leverage this workload estimation and dynamically change the allocation strategy at the access and Edge networks. Finally, we aim to include other machine learning techniques such as K-means clustering along with unsupervised methods to address the limitations of classifying new and unknown types of IoT devices.

REFERENCES

[1] N. Ivanov. (2019). *Unleashing the Internet of Things With In-Memory Computing—IoT Now—How to Run an IoT Enabled Business*. Accessed: Jul. 7, 2021. [Online]. Available: https://www.iot-now.com/2019/01/17/92200-unleashing-internet-things-memory-computing

[2] S. C. Mukhopadhyay and N. K. Suryadevara, "Internet of Things: Challenges and opportunities," in *Internet of Things*. Springer, 2014, pp. 1–17, doi: 10.1007/978-3-319-04223-7_1.

[3] F. Saeik, M. Avgeris, D. Spatharakis, N. Santi, D. Dechouniotis, J. Violos, A. Leivadeas, N. Athanasopoulos, N. Mitton, and S. Papavassiliou, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Comput. Netw.*, vol. 195, Aug. 2021, Art. no. 108177, doi: 10.1016/j.comnet.2021.108177.

[4] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017, doi: 10.1109/COMST.2017.2745201.

[5] D. Dechouniotis, N. Athanasopoulos, A. Leivadeas, N. Mitton, R. Jungers, and S. Papavassiliou, "Edge computing resource allocation for dynamic networks: The DRUID-NET vision and perspective," *Sensors*, vol. 20, no. 8, p. 2191, Apr. 2020, doi: 10.3390/s20082191.

[6] Q. Xu, R. Zheng, W. Saad, and Z. Han, "Device fingerprinting in wireless networks: Challenges and opportunities," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 1, pp. 94–104, 1st Quart., 2016, doi: 10.1109/COMST.2015.2476338.

[7] O. N. Osterbo, D. Zucchetto, K. Mahmood, A. Zanella, and O. Grondalen, "State modulated traffic models for machine type communications," in *Proc. 29th Int. Teletraffic Congr. (ITC)*, Ilmenau, Germany, Sep. 2017, pp. 1–5.

[8] M. Laner, N. Nikaein, P. Svoboda, M. Popovic, D. Drajic, and S. Krco, "Traffic models for machine-to-machine (M2M) communications: Types and applications," in *Machine-to-Machine (M2M) Communications: Architecture, Performance and Applications*, C. Antón-Haro and M. Dohler, Eds. Sawston, U.K.: Woodhead Publishing, 2020, pp. 133–154.

[9] A. Orrevad, "M2M traffic characteristics: When machines participate in communication," Ph.D. dissertation, KTH Inf. Commun. Technol., Stockholm, Sweden, 2009.

[10] M. Miettinen, S. Marchal, I. Hafeez, T. Frassetto, N. Asokan, A.-R. Sadeghi, and S. Tarkoma, "IoT Sentinel demo: Automated device-type identification for security enforcement in IoT," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Atlanta, GA, USA, Jun. 2017, pp. 2511–2514.

[11] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray, and I. Ray, "Behavioral fingerprinting of IoT devices," in *Proc. Workshop Attacks Solutions Hardw. Secur.*, Jan. 2018, pp. 41–50.

[12] Y. Meidan, M. Bohadana, A. Shabtai, M. Ochoa, N. Ole Tippenhauer, J. Davis Guarnizo, and Y. Elovici, "Detection of unauthorized IoT devices using machine learning techniques," 2017, *arXiv:1709.04647*. Accessed: Jul. 27, 2021.

[13] S. Aneja, N. Aneja, and M. S. Islam, "IoT device fingerprint using deep learning," in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IOTAIS)*, Nov. 2018, pp. 174–179.

[14] N. Apthorpe, D. Reisman, and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," 2017, *arXiv:1705.06805*. Accessed: Jul. 27, 2021.

[15] R. Lippmann, D. Fried, K. Piwowarski, and W. Streilein, "Passive operating system identification from TCP/IP packet headers," in *Proc. ICDM Workshop Data Mining Comput. Secur. (DMSEC)*, 2003, pp. 1–10.

[16] J. Kotak and Y. Elovici, "IoT device identification using deep learning," in *Proc. 13th Int. Conf. Comput. Intell. Secur. Inf. Syst. (CISIS)*, 2020, pp. 76–86.

[17] A. Hameed, J. Violos, N. Santi, A. Leivadeas, and N. Mitton, "A machine learning regression approach for throughput estimation in an IoT environment," in *Proc. 14th IEEE Int. Conf. Internet Things*, Melbourne, VIC, Australia, Dec. 2021, pp. 29–36.

[18] M. R. P. Santos, R. M. C. Andrade, D. G. Gomes, and A. C. Callado, "An efficient approach for device identification and traffic classification in IoT ecosystems," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jun. 2018, pp. 304–309.

[19] A. Abdellah, V. Artem, A. Muthanna, D. Gallyamov, and A. Koucheryavy, "Deep learning for IoT traffic prediction based on edge computing," in *Proc. Int. Conf. Distrib. Comput. Commun. Netw.*, Moscow, Russia, 2020, pp. 18–29.

[20] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar, "IoT devices recognition through network traffic analysis," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2018, pp. 5187–5192, doi: 10.1109/BigData.2018.8622243.

[21] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, pp. 18042–18050, 2017, doi: 10.1109/ACCESS.2017.2747560.

[22] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer, and Y. Elovici, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in *Proc. Symp. Appl. Comput. (SAC)*, Marrakech, Morocco, Apr. 2017, pp. 506–509.

[23] 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 Trans. Mobile Comput.*, vol. 18, no. 8, pp. 1745–1759, Aug. 2019, doi: 10.1109/tmc.2018.2866249.
[24] A. Hameed and A. Leivadeas, "IoT traffic multi-classification using network and statistical features in a smart environment," in *Proc. IEEE 25th Int. Workshop Comput. Aided Modeling Design Commun. Links Netw. (CAMAD)*, Pisa, Italy, Sep. 2020, pp. 1–7.
[25] J. Ren, D. J. Dubois, D. Choffnes, A. M. Mandalari, R. Kolcun, and H. Haddadi, "Information exposure from consumer IoT devices: A multidimensional, network-informed measurement approach," in *Proc. Internet Meas. Conf.*, New York, NY, USA, Oct. 2019, pp. 267–279.
[26] C. Zong, R. Xia, and J. Zhang, "Text representation," in *Text Data Mining*, 1st ed. Singapore: Springer, 2021.
[27] J. Brownlee, "How to choose a feature selection method for machine learning," *Mach. Learn. Mastery*, 2020. Accessed: Jul. 27, 2021. [Online]. Available: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/
[28] K. Backhaus, B. Erichson, S. Gensler, R. Weiber, and T. Weiber, "Logistic regression," in *Multivariate Analysis*, K. Backhaus, B. Erichson, S. Gensler, R. Weiber, and T. Weiber, Ed. Wiesbaden, Germany: Springer, 2021, pp. 267–354.
[29] M. Henry, "Review on gradient descent algorithms in deep learning approaches," *J. Innov. Develop. Pharmaceutical Tech. Sci.*, vol. 4, no. 3, pp. 91–95, 2021.
[30] M. Okwu and L. Tartibu, "Artificial neural network," in *Metaheuristic Optimization: Nature-Inspired Algorithms Swarm and Computational Intelligence, Theory and Applications*, M. Okwu and L. Tartibu, Eds. Cham, Switzerland: Springer, 2021, pp. 133–145.
[31] *Scikit Learn, Neural Network Models (Supervised)*. Accessed: Jul. 27, 2021. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html
[32] (2020). *Keras Tuner*. Accessed: Jul. 27, 2021. [Online]. Available: https://keras-team.github.io/keras-tuner/
[33] University of New SouthsWales. *IoT Traffic Traces*. Accessed: Jul. 27, 2021. [Online]. Available: https://iotanalytics.unsw.edu.au/iottraces

AROOSA HAMEED received the master's degree in computer science from Quaid-i-Azam University, Islamabad, Pakistan, in 2018. She is currently pursuing the Ph.D. degree with the Department of Software and Information Technology Engineering, Ecole de Technologie Superieure (ETS), Montreal. Her main research interests include the Internet of Things (IoT), traffic analytics, the IoT services, the IoT security, and machine learning.

JOHN VIOLOS was a Research Associate at the National Technical University of Athens, a Sessional Lecturer at the Harokopio University of Athens, and a Visiting Lecturer at the National and Kapodistrian University of Athens. He was a member of the European Commission's Digital Single Market working group on the code of conduct for switching and porting data between cloud service providers. He is currently a Research Associate with the Department of Software Engineering and Information Technology, ETS. His research interests include deep learning, machine learning, and cloud and edge computing.

ARIS LEIVADEAS (Senior Member, IEEE) received the Diploma degree in electrical and computer engineering from the University of Patras, Greece, in 2008, the M.Sc. degree in engineering from King's College London, U.K., in 2009, and the Ph.D. degree in electrical and computer engineering from the National and Technical University of Athens, Greece, in 2015. From 2015 to 2018, he was a Postdoctoral Researcher with the Department of Systems and Computer Engineering, Carleton University, Ottawa, ON, Canada. In parallel, he worked as an Intern at Ericsson and then at Cisco, Ottawa. He is currently an Associate Professor with the Department of Software and Information Technology Engineering, Ecole de Technologie Superieure (ETS), University of Quebec, Canada. His research interests include cloud computing, the IoT, and network optimization and management. He received the Best Paper Award in ACM ICPE 2018 and IEEE iThings 2021 and the Best Presentation Award in IEEE HPSR 2020.