Received: 18 Aug 2025 | Revised: 22 Nov 2025 | Accepted: 01 Jun 2026 | Published: 30 Jun 2026
Volume 4(1) Pages 43-74 DOI: https://doi.org/ 10.56279/jicts.v4i1.422

JICTS
Journal of ICT Systems

SDNCampus: Application Classification Dataset for Software-Defined Enabled Campus Networks

Wilson C. Chanhemo [a, 1], Mustafa H. Mohsini [b], Mohamedi M. Mjahidi [b], Florence U. Rashidi [a]

[a]*Department of Electronics and Telecommunication Engineering, The University of Dodoma, Dodoma, Tanzania*
[b]*Department of Computer Science and Engineering, The University of Dodoma, Dodoma, Tanzania*

[1]Corresponding author
Email: wilson.chanhemo@udom.ac.tz

Funding information

This research is supported by funding from the University of Dodoma under the Staff Postgraduate Training Program.

Keywords

Application classification
Campus network
Deep Learning
IoT
SDN Dataset

Abstract

The exponential growth and sophistication of campus networks tailored with advanced applications has resulted into network congestion and Quality of Service (QoS) degradation. Meanwhile, the emerging paradigms of Software Defined Networking (SDN) and Deep Learning (DL) are promising technologies that enable network re-design and management in a centralized way. While SDN brings control of the entire network at the controller, DL can leverage statistical information offered by SDN to understand the characteristics of network applications better and inform the controller accordingly. However, deploying SDN coupled with DL in campus networks is still immature due to the lack of a proper dataset to train the DL models. This work presents SDN specific dataset for application classification, including IoT sensors integrated in a campus network. Furthermore, the study demonstrates the usefulness of the proposed dataset by performing experimental evaluation using four (4) time series DL based techniques for traffic classification

1. Introduction

The rapid expansion of networking systems, such as cloud computing, social network applications, and flourishing of communication technologies, such as Internet of Things (IoT), have coincided with the tremendous growth in Internet users over the past few years [1]. In traditional campus networks, devices (e.g., routers or switches) implement the functionality of decision-making processes known as the control plane and routing of network traffic (data plane) independently. Network operators individually configure each device's traffic policies, such as routing, switching and quality of service. The autonomy of each device results in optimization and network administration challenges [2].

Software Defined Networking (SDN) has become a promising technology to solve problems

resulting from conventional networks by enabling flexibility in configuration, decreasing deployment time, providing automation, and simplifying network architecture without requiring knowledge of vendor-specific software and hardware [3]. SDN can control the network from a remote controller with a complete view of the network. Apart from simplified management, SDN can provide network information (data) that can be utilized by state-of-the-art technologies, such as Artificial Intelligence (AI), to understand the hidden characteristics of the applications [4]. By understanding the characteristics of network applications, further optimization can be applied to handle traffic with different Quality of Service (QoS) requirements [5].

Meanwhile, application identification has become essential process to facilitate fine grained management of ever-growing traffic in campus networks. Previous ways of classifying traffic, such as port based methods, deep packet inspection, and conventional machine learning, are becoming less effective due to massive and dynamic network traffic [6]. The shortcomings of previous traffic classification algorithms have raised the need to utilize more advanced technologies, such as Deep Learning (DL), for classification and prediction tasks. DL algorithms can learn complex traffic features and thus accurately classify applications more efficiently than conventional machine learning schemes [7]. Coupled with SDN, DL techniques are expected to revolutionize campus networks and make them smarter in the following ways [8]:

- Streamline the management of underutilized network resources and route traffic intelligently.
- Facilitate smart decisions, carrying out action-oriented procedures, analyzing context information, and obtaining information regarding various applications.
- Provide efficient data traffic visualization to streamline data analysis procedures by creating intelligent algorithms.
- Improve user experience and optimize network performance by assisting SDN to prioritize requests according to user applications and QoS.

Today's campus network is a complicated ecosystem with a variety of user groups, such as students, instructors, administrative staff, and probably some sensor integration for monitoring different conditions [9]. While students consume bandwidth with latency-sensitive applications, such as online gaming, video streaming, and virtual lectures, administrative staff need to run critical services, such as Voice over Internet Protocol (VoIP) and enterprise email systems. Furthermore, researchers need to access cloud-based collaboration tools for research activities using the same infrastructure. These operations generate traffic with conflicting requirements, resulting into straining of network resource and poor service delivery [10]. This situation requires sophisticated algorithms capable of identifying and classifying application flows in real-time to enable precise service delivery.

While SDN and DL technologies can collaborate to facilitate application classification in campus networks, a persistent challenge lies in the availability of realistic SDN datasets to train DL models. Most of existing public dataset have limitations, such as being produced in non-SDN environment, lack of traffic patterns for emerging technologies, including cloud computing, social media applications, and IoT. These limitations have rendered most of existing datasets obsolete [11, 12]. To address the limitations, this study aims to generate a realist SDN dataset for application classification in campus networks. The study aims the following:

(1) To provide extensive analysis of existing public datasets being used to train traffic classification models for deployment in SDN networks. This analysis provides an up-to-date understanding of the strengths and weaknesses of existing datasets that researchers need to be aware of.
(2) To produce ready-to-use and well-balanced public dataset in Comma Separated Values (CSV) format for application classification in SDN campus networks.
(3) To provide a clear roadmap on how other researchers can replicate each subset of the dataset by providing clear steps to be followed during dataset creation process.
(4) To evaluate the created dataset in order to confirm its usability using four (4) time series DL architectures.

2. Publicly Available Application Classification Datasets

Several publicly available datasets have been proposed to support application classification in SDN environments. This section reviews key datasets with a focus on their properties, including the number of extracted features, dataset size, number of classes and algorithms used for evaluation. Understanding these characteristics is essential for assessing their applicability and performance in real-world SDN scenarios. Detailed information regarding each dataset is provided in Table 1.

VPN-nonVPN dataset (ISCXVPN2016): The dataset was prepared by Draper-Gil *et al*. [13] to classify Virtual Private Networks (VPN) and non-VPN traffic. It contains network traffic for various applications organized into seven service groups. VPN traffic represents application traffic routed through a VPN tunnel, whereas non-VPN traffic represents application traffic sent through a non-VPN tunnel. The researchers utilized Wireshark and TCPdump to collect network traces from user accounts registered to use the listed services, such as Skype and Facebook.

Moore dataset: Prepared by Moore *et al*. [14] at Cambridge University labs. The dataset is a collection of network traffic generated by different applications, such as File Transfer Protocol (FTP) and Internet Message Access Protocol (IMAP). The collected network traces were mapped into ten (10) network groups for classification purposes. The dataset consists of 249 extracted statistical features which can be used for training classification algorithms.

Tor-nonTor dataset (ISCXTor2016): Prepared by Lashkari *et al.* [15] at the Canadian Institute for Cybersecurity (CIC). The dataset aims to classify traffic generated by the Tor browser and other non-Tor applications. Three user accounts were created for the browser traffic collection and two users for communication parts, such as chat, mail, File FTP, and peer-to-peer (P2P). Researchers used benign traffic from the ISCXVPN2016 dataset and created seven traffic classes for Tor traffic.

MAWI dataset: Dataset available at Measurement and Analysis on the WIDE Internet (MAWI) [16]. Researchers at MAWI working group have been measuring, analyzing, evaluating and verifying network traffic since the Widely Integrated Distributed Environment (WIDE) Project's inception. The group has been collecting real network traffic since 2000 to date over the WIDE network at different sample points. The collected traces include benign and malicious traffic for numerous applications. The main WIDE's goal is to use the knowledge gained through network operations and to evaluate research results under actual traffic.

Center for Applied Internet Data Analysis (CAIDA) Anonymized Internet Traces dataset: The network traces were collected by Maheshwari *et al*. [17] at various CAIDA's high-speed monitors on high-speed Internet backbone links between April

2008 and January 2019. These datasets are useful for research into the features of internet traffic, including application breakdown, security incidents, geographic and topological distribution, flow volume and duration.

University of Brescia (UNIBS-2009) dataset: The traces were acquired on the edge router of the campus network of the University of Brescia for three consecutive working days by UNIBS [18]. Most of the collected traffic is TCP, making up about 99% of the trace files. The UNIBS-2009 dataset contains packet-level network traffic traces featuring detailed flow records with different attributes, such as Internet Protocol (IP) addresses, port numbers, transport protocols, timestamps, packet sizes, and bidirectional flow statistics. It is useful for performance evaluation of network monitoring algorithms in conventional and academic network environments.

University of Jinan (UJN) dataset: Network traffic traces were collected for two consecutive days at the UJN in 2013 using a Traffic Labeler (TL) developed by Peng *el al*. [19]. Researchers cleaned data by removing network flows with less than 10 packets. The collected traffic contains browsing, chat, cloud disk, and live updates using Windows host devices. The traces have been appropriately labelled and assessed for their pivotal application.

MIRAGE-2019 dataset: Aceto *et al.* [20] collected Android mobile application real dataset using three devices, namely Xiaomi Mi5, Google Nexus 7, and Samsung Galaxy A5. The three devices were installed with the custom firmware CyanogenMod v13.0 (corresponding to Android version 6.0.1). The dataset was collected in a controlled environment at the ARCLAB laboratory at the University of Napoli. It is useful for mobile traffic analysis.

Anon17 dataset: Between 2014 and 2017, Shahbar and Zincir-Heywood [21] at the Network Information Management and Security (NIMS) lab collected the dataset in a real-world network environment. The dataset was labelled based on information provided by the anonymity services themselves. Researchers used the Tranalyzer tool to extract features from the collected raw data. There are five ARFF datasets for various traffic classes.

DNS over HTTPS (DoH) dataset: The dataset was collected by Jerabek *et al.* [22] at the Brno University of Technology, Czech Republic. Researchers deployed the tcpdump packet capture program to collect the dataset in two environments: (i) generated DoH and HTTPS traffic from a virtualized environment (Docker), and (ii) generated real-world DoH and HTTPS traffic from a real large Internet Service Provider (ISP) network. Real-world collected data portrays the actual network environment, such as packet timing, loss, retransmissions, and connection problems. The purpose of the produced data is to offer heterogeneous DoH traffic for various applications.

CIRA-CIC-DoHBrw-2020 dataset: Montazerishatoori *et al*. [23] collected DNS over HTTPs (DoH) dataset in a controlled environment using simulation at CIC and contained Benign-DoH, Malicious-DoH, and non-DoH traffic. The dataset aims to discover tunneling activities that use DNS communications over HTTPS by presenting a two layered technique for detecting and describing DoH traffic using a time series classifier.

MobileGT dataset: The dataset was generated by mobile apps and collected in a monitored smartphone environment in work by Wang *et al*. [24]. The dataset comprises Active Mobile Traffic Data (AMTD) and Passive Mobile Traffic Data (PMTD). AMTD was gathered using the mgtClient on 10 volunteer smartphones between 2016 and 2017, whereas PMTD was collected in 2018. Its rich feature set and coverage of common mobile applications make it particularly suitable for

research on mobile traffic classification and user behavior analysis.

CESNET-QUIC22: Luxemburk *et al*. [25] created a dataset that contains one-month of Quick UDP Internet Connections (QUIC) traffic collected from 100 Gbps backbone lines of a real network serving approximately half a million people. The network provides realistic characteristics of network traffic originating from various web browsers, operating systems, mobile devices and desktop machines. Because of the organic nature of the collected traffic, the dataset includes rich behaviors (for example, all conceivable user actions and settings) of online services and mobile applications, which would be hard to obtain in a lab-generated dataset.

CIC IoT dataset 2022: Dadkhah *at al*. [26] created CIC IoT dataset 2022, which is made up of state-of-the-art profiling, behavioral analysis and vulnerability testing of various IoT devices with different protocols, such as IEEE 802.11, Zigbee-based, and Z-Wave. The dataset provides a comprehensive foundation for behavioral profiling and intrusion analytics in IoT environments.

SDN Dataset: This dataset was created in a work by Wassie *et al*. [27] to categorize mice and elephant flows in SDN networks. The dataset was generated using a Mininet emulator where network topologies were created to generate traffic. The dataset generation process started with creating ten topologies in Mininet, in which switches are connected to a single Ryu controller. The dataset is used to simulate and detect mice and elephant flows from TCP, UDP, and ICMP traffic.

CESNET-TLS22 dataset: The dataset was built from CESNET2 network traffic observed in the first two weeks of October 2021 by Luxemburk and Čejka [28]. The dataset contains 200 service classes and was formed from the TLS handshake phase of the communication flow. The capture environment provided realistic features of traffic coming from various web browsers, operating systems, mobile devices, desktop computers, and both HTTP/1.1 and HTTP/2 protocols. CESNET-TLS22 provides a way for developing and evaluating models for fine grained service classification in encrypted networking environments.

NetTiSA dataset: In a study by Koumar *at al*. [29], Network Time Series Analysed (NetTiSA) dataset was created. NetTiSA is an enhanced IP flow dataset that contains a universal bandwidth constrained feature vector, created from 15 existing raw data generated to create public datasets. The combined multiclass dataset contains 44 classes with 20 features each. It is helpful for traffic classification where time series features are vital.

VNAT dataset: Jorgensen *et al*. [30] created VNAT dataset which contains labelled Packet CAPture (PCAP) files, both encrypted and unencrypted, generated by ten applications. The data were captured using tcpdump in an isolated network environment to avoid data corruption from other sources. The dataset is intended to support the development of machine learning techniques that allow operators to realize the traffic categories of encrypted and unencrypted applications.

D-OutCsv dataset: The dataset was created by performing a series of actions on apps, most related to social media and email applications on a controlled network segment in a work by Pathmaperuma *et al*. [5]. The entire dataset was formed by sniffing traffic on a wireless LAN using the Airmon-ng and Airodump-ng tools from the Aircrack-ng software suite. Researchers can use the provided dataset to classify in-app activities by observing only a part of their traffic.

Open HTTPS Dataset: Shbair *et al*. [31] created this dataset using Google Chrome and Mozilla Firefox by crawling over 700 HTTPS websites. The scanning method was automated using a local machine and a pre-configured remote proxy that dumped all traffic with port 443. The

PCAP files with full payload were collected, and then the Pkt2flow application was used to extract traffic flows. The dataset is useful for classifying HTTPS traffic in different feature dimensions.

Although the datasets discussed are used for studies involving traffic classification in SDN, there are a number of limitations that still exists: (i) Most were not created using SDN platforms, (ii) Some are very specific to applications and do not cover a wider range of traffic, and (iii) Some use anonymized data that may not reflect realistic data flow logic. Given these limitations, the resulting developed models may present a compatibility issue because the conventional network and SDN are fundamentally different paradigms [5]. Not only that, most of the available public datasets have the incomplete documentation, making it rather difficult to replicate the same or to provide a detailed description of the datasets.

3. Theoretical and Methodological Framework

The study was grounded in the principles of SDN and DL to form the foundation of the created dataset and its evaluation process. SDN separated the control and data planes to enable centralized management and network programmability, while DL provided the basis of data-driven modelling through feature learning. The methodology followed a framework proposed by Sharafaldin *et al*. [32] (Figure 1). Other frameworks also exist in the literature but were considered less effective. For example, Velarde-Alvarado *et al*. [33] proposed a framework to generate dataset based on traffic aggregation, which was not a requirement for this study. Shiravi *et al*. [34] proposed a framework to generate dataset specific to Intrusion Detection Systems (IDS) and was limited to small traffic and device types, while the framework by Ferriyan *et al*. [35] emphasized encrypted traffic only. The framework by Sharafaldin *et al*. [32] adopted in this study ensured systematic and stepwise testbed setup, traffic capture, labeling of the dataset and evaluation. Furthermore, the framework is well documented for use by research community. The framework has 11 features:

(1) Complete Network Configuration: The study deployed a complete network with both real and virtualized resources representing the study's main entities.
(2) Complete Traffic: During experiments, a complete communication of services from clients and servers was ensured.
(3) Labelled Dataset: Each application was run separately to ensure a correct and uncorrupted dataset.
(4) Complete Interaction: Traffic interaction for both LAN and WAN was configured to ensure realistic network characteristics.
(5) Complete Capture: The capturing process grabbed all traffic intended for the study and removed none.
(6) Available Protocols: The proposed dataset incorporated a wide range of protocols with traffic of different characteristics to represent realistic network behaviour.
(7) Traffic Diversity: The proposed dataset included a wide range of applications with traffic of different characteristics.
(8) No Anonymity: The dataset produced in this study resulted from raw data files without anonymization.
(9) Heterogeneity: Different traffic sources such as smartphones, IoT sensors, and computers were used for data collection.
(10) Feature set Generation: To generate a reliable feature set, the study deployed CICFlowMeter tool developed by CIC [36].

Metadata: This study's dataset is made available to the publice with complete information on how it was obtained, the production testbed used and the hardware/software deployed.

Table 1. Existing public datasets.

SN	DATASET	Raw Data Size	Features	Classes	Traffic Sources	Year	labelled	Format	Production Settings	SDN-Environment	Testing Algorithms	Classification Accuracy
1	ISCXVPN2016	24.639GB	23	14	HTTPS, SMPTS, POP3, IMAPS, ICQ, AIM, Skype, Facebook, Hangouts, FTPS, SFTP, uTorrent, BitTorrent	2016	Yes	PCAP and ARFF	Real traffic Generated by Lab Members	NO	C4.5 and KNN	Above 80% on average
2	Moore	Varies	249	12	SMTO, Web POP3, FTP, DNS, BitTorrent, MySQL, Virus, Telnet, WOW Windows media player	2005	Yes	ARFF	Real network traffic collection	NO	Naive Bayes (NB)	65% accuracy for data from the same period. 95% when combined with a number of simple refinements
3	Tor-nonTor (ISCXTor2016)	22GB	23	8	HTTPS, SMPTS, POP3, IMAPS, ICQ, AIM, Skype, Facebook,	2016	Yes	CSV and PCAP	Testbed Emulation by Whonix OS	NO	Zero R, Decision Tree (C4.5), KNN, Random Forest (RF)	Above 90% with C4.5

SN	DATASET	Raw Data Size	Features	Classes	Traffic Sources	Year	labelled	Format	Production Settings	SDN-Environment	Testing Algorithms	Classification Accuracy
					Hangouts, FTPS, SFTP, uTorrent, BitTorrent Tor							
4	MAWI	Varies	various	Not classified	various	2000-2024	No	PCAP	Capture from real network	NO	Unknown	Unknown
5	CAIDA Anonymized Internet Traces Dataset	Varies	various	Not classified	various	April 2008 - January 2019	No	packet header only	Capture from real network	NO	Unkown	Unkown
6	UNIBS-2009 dataset	27GB	various	7	Web (HTTP and HTTPS), Mail (POP3, IMAP4, SMTP and their SSL variants, Skype, BitTorrent and Edonkey, FTP, SSH, and MSN	2009	Yes	PCAP	Capture from real network	NO	Unkown	Unkown

SN	DATASET	Raw Data Size	Features	Classes	Traffic Sources	Year	labelled	Format	Production Settings	SDN-Environment	Testing Algorithms	Classification Accuracy
7	UJN dataset	2.12GB	Various	8	Browsing, chat, cloud disk, live update, stream media, mail, ftp, p2p, other	2013	Yes		Capture from lab settings	NO	Tested with port based methods only	100% with port based testing method
8	MIRAGE-2019	1.4GB	102	40	40 Android apps	2017-2019	YES	JSON	Real Data on controlled laboratory environment	NO	Unkown	Unknown
9	Anon17	Unkown	79	24	Various sources	2014-2017	YES	ARFF	Capture from real network	NO	Unknown	Unknown
10	DNS over HTTPS (DoH) dataset	430 GB	16	2	Firefox and Chrome browsers	2021		PCAP, CSV	Capture on Virtual and real networks	NO	Unknown	Unknown
11	CIRA-CIC-DoHBrw-2020 dataset	75.3GB	28	2	Mozilla Firefox, Chrome, dns2tcp, DNSCat2, and Iodine	2020	YES	PCAP, CSV	Simulated environment	NO	RF, C4.5, SVM (SVM), NB DNN CNN	More THAN 99% for statistical features, 98% for time series features
12	MobileGT dataset	14.7GB	18	12	12 Mobile apps	2016-2018	YES	ARFF	Capture from real network	NO	C4.5, KNN, BayesNet, RF	C4.5 achieved 91%

SN	DATASET	Raw Data Size	Features	Classes	Traffic Sources	Year	labelled	Format	Production Settings	SDN-Environment	Testing Algorithms	Classification Accuracy
											and AdaBoost	
13	CESNET-QUIC22	27TB	29	various	Various sources	2022	NO	CSV	Capture from real network	NO	Unknown	Unknown
14	CESNET-TLS22	42GB	29	200	Various sources	2022	YES	CSV	Capture from real network	NO	1D-CNN, LightGBM	Over 97% with 1D-CNN
15	CIC IoT dataset 2022	51.42GB	48	3	Various IoT devices	2022	YES	PCAP,CSV	Controlled Laboratory environment	NO	RF. Guassian NB, DT, LDA, AdaBoost, Ridge, Perceptron, Passive-Aggressive, XGBoost, KNN, LinearSVC SGD	98.5% with RF
16	**SDN Dataset**	Unknown	23	2	Mininet virtual hosts	2020	YES	CSV	Simulation on Mininet	YES	XGBoost, GBM and DRF	100% with DRF
17	**NetTiSA**	varies	20	44	Various	2023	YES	CSV	Generated from various	NO	XGBoost	98%

SN	DATASET	Raw Data Size	Features	Classes	Traffic Sources	Year	labelled	Format	Production Settings	SDN-Environment	Testing Algorithms	Classification Accuracy
									public raw datasets			
18	**VNAT**	36.1GB	18	5	Vimeo, Netflix, YouTube, Zoiper, Skype, SSH, RDP, SFTP, RSYNC, SCP	2024	YES	PCAP	Capture from isolated network segment	NO	out-of-distribution" (OOD) score	98%
19	**D-OutCsv dataset.**	Unkown	48	92	Facebook, Instagram, YouTube, WhatsApp, Viber, Gmail, Skype, Messenger,	2022	YES	CSV	Capture on real controlled environment	NO	1D CNN, RNN, SAE DNN	93% with DNN
20	**Open HTTPS Dataset**	53.3GB	various	Unkown	Chrome, Firefox	2016	NO	PCAP	Capture on real network	NO	Unkown	Unkown

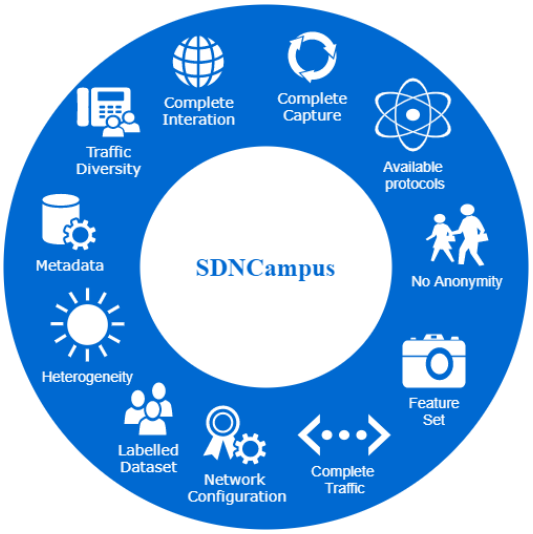


Figure 1. SDNCampus dataset properties.
(Source: Adopted and compiled from Sharafaldin *et al* [32].)

Using the adopted framework, the SDNCampus dataset created in this study have properties summarized in Figure 1. These properties ensure that the dataset addresses the limitations of the current existing datasets by (i) being created in realistic SDN environment, (ii) being generic to multiple and mixed applications, and (iii) being able to avoid anonymization by creating the dataset from raw captured network traffic.

4. Materials and Methods

This section introduces the SDNCampus dataset generation process. An in-depth description of all steps and resources involved in producing this dataset is explained. First, the network setup for running the experiments is described followed by raw data (PCAP format) collection procedures. The procedures to generate the final dataset in CSV file format are explained followed by the dataset pre-processing, DL algorithm selection, and hyperparameters tuning. Finally, the choice of classification metrics and model training procedures are explained.

a. Network Setup for Data Collection

Figure 2 shows the constructed network architecture for data collection. The network was constructed using two desktop computers, PVE1 and PVE2. Both computers were equipped with 1 terabyte of storage, 16 gigabytes of RAM, and Core i7 processor. The two servers were running Proxmox hypervisor version 8.1. Proxmox is an open-source server management tool designed for virtualization. Proxmox tightly integrates software defined storage, networking and Kernel based Virtual Machine (KVM) hypervisor with Linux Containers (LXC). Each server hosted several Virtual Machines (VMs) (Table 2). In each Proxmox server, Open vSwitch (OVS) version 3.1.0 was installed and virtual bridges vmbr1 and vmbr2 were created. To enable communication from LAN housed on the virtual environment of the servers and physical devices to the campus LAN, internet, and between the two servers, one physical Network Interface Card (NIC) was added to the vmbr1 bridge on each server. The second bridge, vmbr2, of each server was connected to the second NIC card of the server to bridge each server's LAN to the physical world via the campus LAN.

For each VM or physical device running Ubuntu or any Linux distribution, OVS was installed and configured to use the OVS bridge for communication instead of the traditional IP stack. This was done to maximize the use of SDN in the experiment. The use of OVS adaptors differentiates this setup from past studies where the traditional use of the IP stack was usually applied. Ryu controller (Version 4.34) installed on VM101 in PVE1 was configured to manage each OVS created on PVE1, PVE2 and Raspberry Pi. Both the controller and OVS were configured to run OpenFlow version 1.3, which is considered the most stable.

From PVE1, a local network was extended to an access point named WLAN-01 using vmbr2. WLAN-01 was configured to accept connections from authenticated devices for data collection

activity. PVE2 LAN was also extended to WLAN-02 to connect additional devices. PVE1 and PVE2 were connected on two different VLANs within the campus networks, so their communications had to pass through the backbone of the campus network for inter-vlan communication. This setup allowed communication for testbed hosts to be influenced by realistic network conditions, such as throughput, latency, delay, and jitter. The Pfsense firewall was set up to filter out unnecessary traffic whenever necessary. Filtering was also performed on client machines, such as computers and smartphones, to allow only required applications to run. The entire list of devices and software used to set the network for data collection is shown in Table 2.

b. Data Collection Procedures

Raw data was collected using tcpdump, available on many Linux distributions. The experiment conducted involved running selected popular applications and sensors for IoT (Table 3). The selection of applications was done to ensure (i) a variety of protocols is obtained, (ii) both resource-hungry and moderate applications are involved in the experiment, (iii) social networking applications are involved due to their growing trend, (iv) both local and remote (internet based) services are included, and (v) elephant and mice network traffic flows are generated in the network. For each application, one or more pcap files were collected so that enough traffic flows would be generated. A step-by-step collection of each application's PCAP data is shown in Table 4.

Next, the PCAP files were processed using CICFlowmeter. CICFlowmeter is a network traffic flow generator that has been written in java at CIC. CICFlowmeter generates more than 80 statistical traffic flow features from the input PCAP file in the form of csv file. Despite the fact that there are other tools in the literature like Argus and Bro-IDS, CICFlowMeter was a better choice because none of the other tools consider time-based aspects of applications into account. Description of each statistical feature generated by CICFlowmeter can be found at CIC-UNB [36].

The socket information (IP address, port numbers) ware removed from the CSV data as part of cleaning because they are likely to change from one network to another and, hence, is not an essential traffic feature. After removing the socket information, the CSV files remained with 77 features. Since each PCAP file was generated from a single application, the process for labelling flows was done using a Python script. For each application stated in Table 3, we extracted 100,000 flows, making 3 million flows of the clean dataset in CSV format.

c. Dataset Pre-processing

The pre-processing stage was done to ensure that the dataset was suitable for the model training phase. The following steps were taken for each flow entry in the dataset:

- To limit the scale of the feature values, each feature was standardized using a StandardScaler available in Sci-Kit Learn package. StandardScaler uses Equation (1) to standardize each feature to a uniform scale, where Z is the Z-score of the value X_i, calculated using the mean and standard deviation of feature X.

$$Z = ((X_i\text{-Mean}(x)))/(\text{stdev}(x)) \quad (1)$$

- The labelled strings were converted to numerical values using a Python script, and thereafter, the one-hot encoding technique was used to convert numerical labels to corresponding binary values.
- The evaluation used 100,000 instances (rows) of each application for individual application classification, 100,000 instances for each class for application

group classification, and 50,000 instances of each class in binary classification.
- The dataset was separated for training, testing, and evaluation in a ratio of 80%, 10%, and 10%, respectively.

d. Deep Learning Algorithm Selection

Any practical modelling task requires the selection of a model. The objective of model selection is to identify the best approximation among a set of candidate models [37]. DL algorithms like Multilayer Perceptron (MLP), Stacked Autoencoder (SAE), and Recurrent Neural Network (RNN) variations, such as Long Short-Term Memory (LSTM), are frequently utilized for classifying network traffic [7]. The proposed dataset in this study reveals a chronological progression in communication patterns over time. Hence, algorithms that capture the time relationship were considered appropriate for evaluation purposes. For this reason, RNN variants, namely LSTM, Bidirectional LSTM (BiLSTM), Gated Recurrent Unit (GRU), and Bidirectional GRU (BiGRU) were chosen. The advantage of the chosen variants over simple RNN is their ability to learn longer relationships between data samples [38].

LSTM detains long-term dependencies in data. Utilizing a memory cell and gates (forget, input, output, memory) to regulate input flow. LSTMs can circumvent the vanishing gradient issue that befalls conventional RNNs. Unlike standard LSTM, the input in BiLSTM flows in both directions. BiLSTM adds one more LSTM layer, reversing the direction of information flow. BiLSTM improves upon a standard LSTM by processing sequence data in both forward and backward directions to enable the model to capture dependencies from both past and future contexts simultaneously.

GRU has a streamlined design with two gates: update gate and reset gate. The update gate controls how much of the prior hidden state should be retained, and the reset gate regulates how much past information should be forgotten. Since it lacks the forget gate, it has fewer parameters compared with standard LSTM. Therefore, GRU is computationally simpler and trains faster than an LSTM [38]. BiGRU is a sequence processing model that consists of two GRUs, one takes the input in a forward path and the other in a backward path. The combined outputs of the forward and backwards GRUs result in a richer representation of the input sequence.

e. Traffic Classification Metrics

To evaluate the proposed algorithms' performance, the study applied Accuracy (AC), Recall (RC), Precision (PR), and F1 Score (F1) metrics. The four metrics are described mathematically in equation (2), (3), (4), and (5), where TP, TN, FP, and FN stand for True Positive, True Negative, False Positive, and False Negative, respectively:

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \quad (2)$$

$$RC = \frac{TP}{TP + FN} \quad (3)$$

$$PR = \frac{TP}{TP + FP} \quad (4)$$

$$F1 = 2 \times \frac{RC\ x\ PR}{RC + PR} \quad (5)$$

f. Model Training Environment

The workflow in Figure 3 with grid search algorithm was applied to search among a combination of parameters, especially input layer size, learning rate, dropout, batch size, and hidden layer size. Because of the intensive resource requirement (memory and computing) for implementing the grid search algorithm, it was implemented on Ubuntu computer with 256 GB RAM and a 32-core processor. To avoid searching across a large number of parameters, grid search

was preceded by random search to confine the search to a scope of parameters. After obtaining the optimal parameters stated in Table 5, the final models were trained on a personal computer with 16 GB RAM and a Core i7 processor with 2 cores, demonstrating that the models were optimal and can be trained in a less-resourced computing environment.

g. Training Procedures and Feature Processing

The four models, LSTM, BiLSTM, GRU, and BiGRU were trained using 77 statistical features of the datasets generated by CICFlowmeter. The dataset was divided into 80/10/10 for training, validation, and testing, respectively. StandardScaler was used to normalize features into a uniform scale. Each model used two recurrent layers followed by batch normalization and dropout to improve generalization. A Global Max-Pooling 1D layer was added right before the dense output layer to extract the most relevant temporal activations. The training employed standard Adam optimizer with early stopping monitored validation loss to retain best performing checkpoint. The entire procedure is shown by Algorithm 1.

Algorithm 1:

Training and Evaluation of LSTM, BiLSTM, GRU and BiGRU Models

Inputs:

Labeled flow dataset D (77 standardized features)

Split ratios: 80/10/10 → (train, validation, test)

Procedure:

1. Preprocess data

– Standardize each feature (zero mean, unit variance)

– Split D into train/val/test with stratification

2. Build model

– Input layer: 77 features

– Recurrent Layer 1

if model ∈{LSTM, BiLSTM}:

activation = sigmoid

recurrent_activation = tanh

if model ∈{GRU, BiGRU}:

activation = tanh

recurrent_activation = sigmoid

– Batch Normalization

– Dropout

– Recurrent Layer 2 (same variant and activations)

– Batch Normalization

– Dropout

– Global Max-Pooling 1D

– Dense output layer (Softmax)

3. Compile model

– Optimizer: Adam

– Loss: categorical cross-entropy

– Metrics: Accuracy, Precision, Recall, F1

4. Train model

– Apply early stopping based on validation loss

– Restore best model weights

5. Evaluate model

– Compute test loss and metrics on test set

– Report Accuracy, Macro-F1, Precision, Recall,

End Algorithm

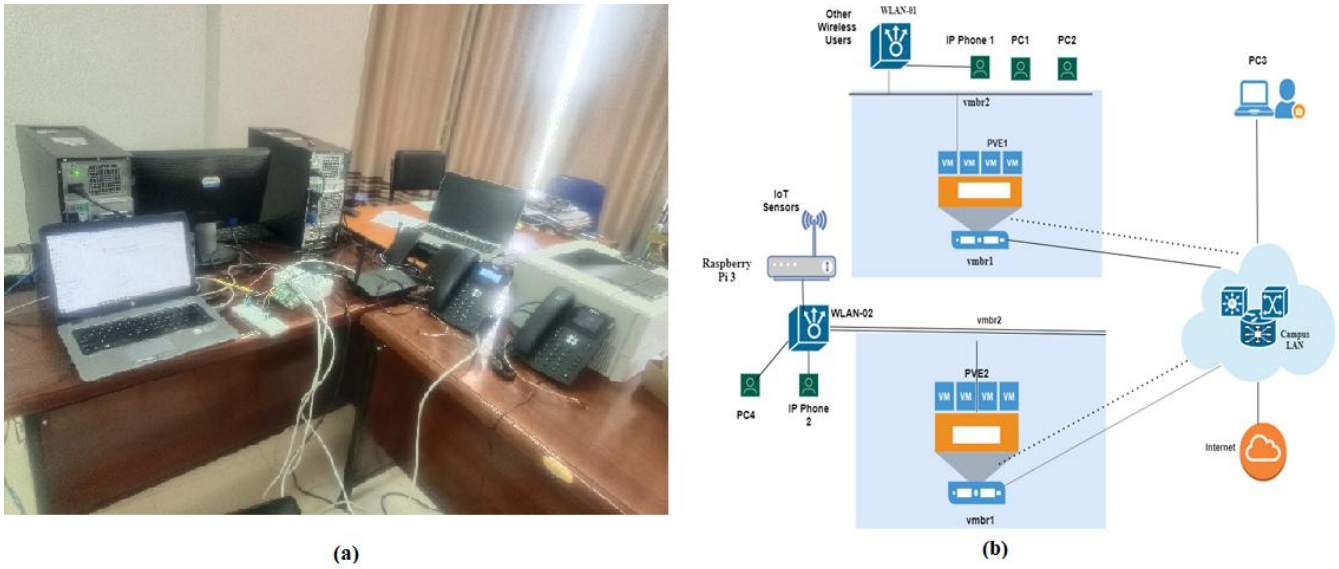


Figure 2. Network setup.

Table 2. Network setup details.

Device	Virtual Machines	Operating System/Device Name	Installed Applications	Purpose
PVE1	VM-100	PfSense Firewall	PfSense Firewall	Filter undesired traffic, Site-to-Site VPN setting , Remote SSL VPN setting, L2TP setting
	VM-101	Ubuntu 22.04	Ryu controller, MariaDB Database Software	Manage Open vSwitches, , to store sensor readings for IoT devices
	VM-102	Sangoma Linux 7 (Core)	FreePBX 15.0.23	Handle VoIP calls hosted and remote extensions on the other VoIP system using SIP trunking.
PVE2	VM-100	PfSense Firewall	PfSense Firewall	Filter undesired traffic, Site-to-Site VPN setting
	VM-101	Ubuntu 22.04	SSH Server daemon, Samba File server	Handle file transfer between computers, Handle remote connections
	VM-102	Sangoma Linux 7 (Core)	FreePBX 15.0.23	Handle VoIP calls hosted and remote extensions on the other VoIP system using SIP trunking.
WLAN-01	--	Dlink-920-caa2	---	To Provide wireless and wired access to users of the testbed network.
WLAN-02	---	TP-Link Archer C24	--	To Provide wireless and wired access to users of the testbed network
PC1	--	Windows 10	FileZilla Client, Putty software,	Manage file transfers, handle remote management via SSH client connection, make ,VoIP

Device	Virtual Machines	Operating System/Device Name	Installed Applications	Purpose
			3CX Softphone, µTorrent, BitTorrent, TeamViewer, Anydesk	calls to other endpoints, initiate windows update activity.
PC2	---	Ubuntu 22.04	rsync	File transfer and backup management, File server
IP Phone 1	--	Fanvil X3SG Pro	---	To make VoIP calls
IP Phone 2	--	Fanvil X3SG Pro	---	To make VoIP calls
PC3	----	Windows 10	OpenVPN connect,	For remote SSL access to tesbed LAN
Temperature and Humidity Sensor	---	-----	Temperature and Humidity sensor	Read air humidity and temperature in a given area
Motion Sensor	---	----	Motion sensing	Sense movements within the testbed area
Raspberry Pi 3	---	Linux raspberrypi 6.6.31+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.6.31-1+rpt1 (2024-05-29) aarch64 GNU/Linux	Motion, Temperature and Humidity monitor applications	Send sensor reading to a MariaDB database
PC4	---	Windows 10	, µTorrent, BitTorrent, TeamViewer, Anydesk	P2P traffic generation, Remote connection services provision.

Table 3. Applications selected for dataset generation.

Application Group	Application Name	Service Location
Social networking	Facebook, X (Twitter), Facebook Messenger, SnapChat, Telegram, TikTok, Instagram, WhatsApp	Internet
Streaming Applications	YouTube, Netflix, Spotify	Internet
Remote management	SSH, Team Viewer, Anydesk, Remote Desktop Connection (RDP)	Internet and LAN
Mail Clients	Microsoft outlook, Gmail application	Internet
File transfer	Google Drive, Dropbox, FileZilla	Internet and LAN
Torrent Applications	UTorrent, BitTorrent	Internet
VoIP Communication	3CX phone, Mizudroid, Fanvil X3SG Pro	LAN
Video conference	Zoom, Google Meet	Internet
VPN	IPSec, OpenVPN, L2TP	LAN and Internet
IoT logging	Temperature and humidity sensor, Water Sensor	LAN

Table 4. Traffic capture procedures.

Application Group	Capture procedure
Social networking and streaming applications	1. At least five (10) volunteers were requested to interact with social networking applications installed on smart phones. Sessions were carried out in different time and day in order to capture network variations such as the busy and non-busy hours 2. Volunteers were connected on WLAN-01 while interacting with the applications. 3. Volunteers were allowed to interact with single application in order to avoid noisy during capturing of a specific application. 4. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1
mote Connection	1. Installed SSH and configured VM101 in PVE2 to allow remote connection from PC1 running Putty software. Then, using Putty a series of activities relating to Ubuntu server management such as software update, user and file system management etc was performed. 2. Installed TeamViewer and Anydesk on PC1 and PC4. PC1 and PC4 are connected to internet via Pfsense firewall attached to campus LAN via OVS port on PVE1 and PVE2 respectively. 3. PC1 cannect to PC4 via TeamViewer and Anydesk, one application at a time and perform various management tasks related to windows management. 4. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1.
Mail Applications	1. Users connected to WLAN-01 were requested to send emails using Outlook and Gmail clients installed on their smart phones. Emails were sent in mixed ways such as those with or without attachments. 2. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1.
File transfer	1. Random files (text document, picture and videos) were prepared to synchronize between PC1 to the internet storage (google drive and Dropbox) by using two (2) authentic user account. 2. Local file transfer was performed by transferring files between PC1 and VM101 on PVE2. In this case VM101 was configures to accept Secure File Transfer Protocol (SFTP) and FileZilla client was installed on PC1 to facilitate file transfer between the two machines. 3. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1.
Torrent Applications	1. P2P applications (µTorrent and BitTorrent) were installed on PC1, PC2 and some volunteer devices connected on WLAN-01. Then each device was allowed to turn on the application for downloading some files on the internet. The installed applications were running with default settings to allow father and son peers relationship. 2. Raw data is captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1
VoIP Communication	1. FreePBX was installed on PVE1 and PVE2 as voice servers to handle calls for extensions on local voice server and to perform trunking calls between them. 2. Extensions were created in both servers and both soft phones (3CX and Mizudroid) and physical phones were added to the systems. 3. Calls were made for extensions within single voice server and between the two servers using SIP protocol.

Application Group	Capture procedure
	4. Extensions on WLAN-01 and WLAN-02 are configured to connect to voice servers on PVE2 and PVE1 in order to have single point of data capture for both within and outside the voice server calls. 5. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1.
Video conference	1. At least five (10) remote volunteers were asked to join random video conference meetings with one researcher who was using PC1 on the testbed. There were five (5) sessions, each lasting for about 40 minutes. Sessions were carried out in different time and day in order to capture network variations such as the busy and non-busy hours. 2. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1.
VPN	1. Site to Site (S2S) VPN was configured between the two Pfsense firewalls on PVE1 and PVE1 to connect the internal subnets of the two PVE servers. 2. Any communication between hosts On PVE1 and PVE2 sides is carried via this S2S tunnel. 3. For Remote access VPN, OpenVPN service was configures on Pfsense firewall on PVE1 and OpenVPN Connect client was installed on PC3. Connections were then established from PC3 to the internal LAN in PVE1 where various management tasks were performed to generate network traffic. 4. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1
IoT applications	1. Temperature and humidity sensor (DHT11)) and water sensor (Ampere Electronics 3-pin water sensor) were connected to Raspberry Pi 3 one at a time. 2. Python scripts dht11.py and waterSensor.py were created and loaded on Raspberry Pi for controlling and reading the sensor inputs one at a time. Sensing was conducted for 3 days until enough captured data was obtained. 3. The Raspberry Pi was connected on WLAN-02 using Ethernet cable for reliability purposes for sending sensor data to MariaDB database installed on VM102 on PVE1 4. Raw data was captured on vmbr1 port connecting Pfsense firewall WAN port in PVE1

Table 5. Selected hyperparameters.

Classification Approach	Output Activation Function	Optimization Algorithm	Hidden Layer 1 Units	Hidden Layer 2 Units	Loss Function	Drop out	Learning rate	Batch	Epochs	Sequence Length
Multiclass Classification	Softmax	Adam	128	64	Categorical Crossentropy	0.3	0.002	256	100	1
Binary Classification	Softmax	Adam	64	32	Categorical Crossentropy	0.5	0.001	8	50	1

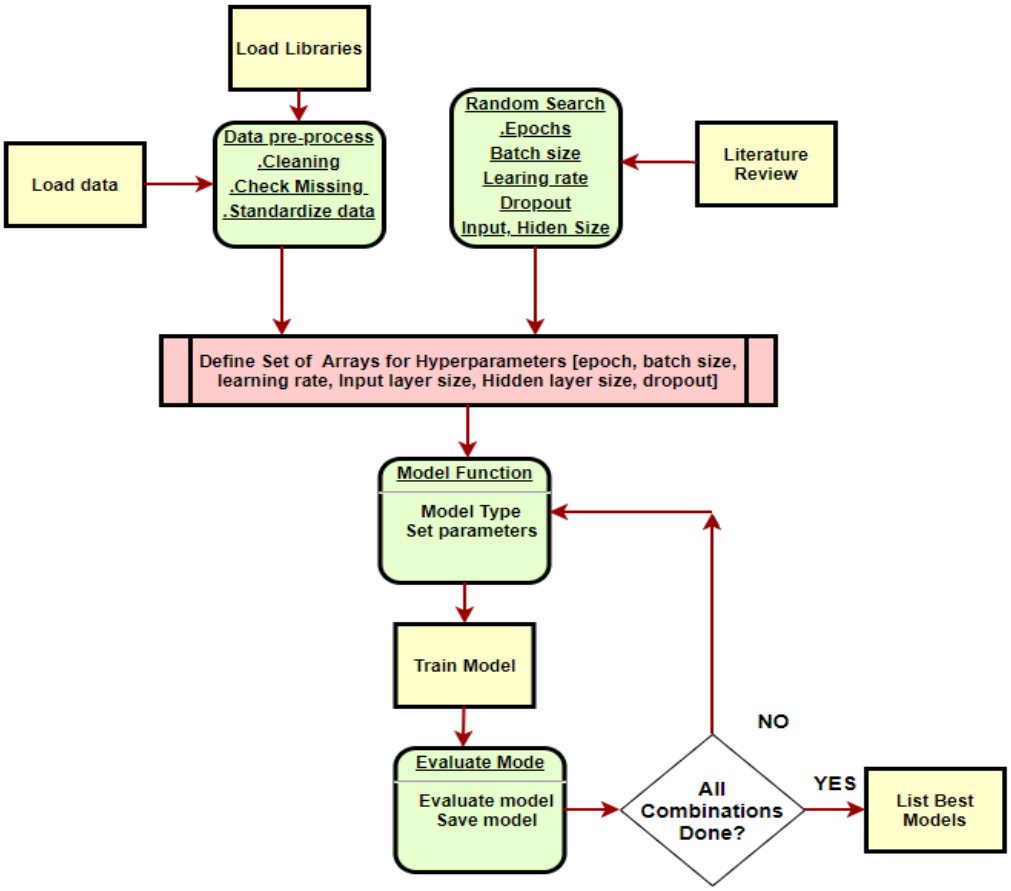


Figure 3. Workflow for hyperparameters tuning and model training.

5. Results

Figures 4, 5, and 6 show the classification performance in three different scenarios: individual, group, and binary classification. The classification performance for individual applications (Figure 4) achieved accuracy slightly above 90% for BiLSTM and BiGRU, while LSTM and GRU models achieved performance slightly above 88%. Further exploration of classification results by looking at the confusion matrix in Table 6 indicates that the models experienced high confusion for classifying social media and torrent applications. In Table 6, the confusion for classifying BitTorrent and uTorrent applications was above 20%, indicating a high similarity of the two applications.

In addition to that, TikTok and X (formerly known as Twitter) presented a confusion result of above 10% in all four models. In all cases, TikTok presented significantly confusing results than other social media applications. The overall performance for individual application classification for all models was affected by the feature confusion mainly from social media applications, which presented notable similarities to the models.

Table 7 shows individual application performance scores where VoIP, OpenVPN, and L2TP presented higher scores (accuracy above 98%) compared with other applications. The notable performance suggests that the network data generated by these applications has distinct features that enable streamlined differentiation from other types of traffic. Their high classification accuracy

score is an outcome of clearly defined patterns in communication procedures, encryption/decryption algorithms, and packet structures. VoIP applications, for example, typically provide steady, low-latency packet streams. However, VPN protocols, such as OpenVPN and L2TP, encapsulate traffic in a manner that is very different from that of regular web or streaming applications.

For application group classification (Figure 5), there was an improvement that reached slightly above 94.6% overall accuracy for BiLSTM and BiGRU, while LSTM and GRU attained an overall accuracy slightly above 93%. The improvement in classification accuracy was a direct result of combining similar applications into groups, which enabled their shared features to be merged. This technique of combining similar applications into one group is standard practice in practical environments where network administrators would like to apply the same network management rules to similar applications. Table 8 shows the detailed results of application group classification. It was clearly shown that applications with unique features, such as VPN, were classified with fewer false positives (precision approximately 98%) as compared to applications with shared features, such as streaming and social media applications which had precision of approximately 89%.

Figure 6 shows the performance for IoT and non-IoT classification. All models attained above 99% overall accuracy and Table 9 illustrates additional performance parameters where in all cases the models scored an overall accuracy above 99%. The notable performance of the models in binary classification was a result of unique traffic features presented by the IoT sensor data collection application, which presents consistent data logging to the database. This feature is different from other user applications, which may be accessing different contents, such as text, pictures, video and voice simultaneously. As a result, models were able to distinguish these applications well.

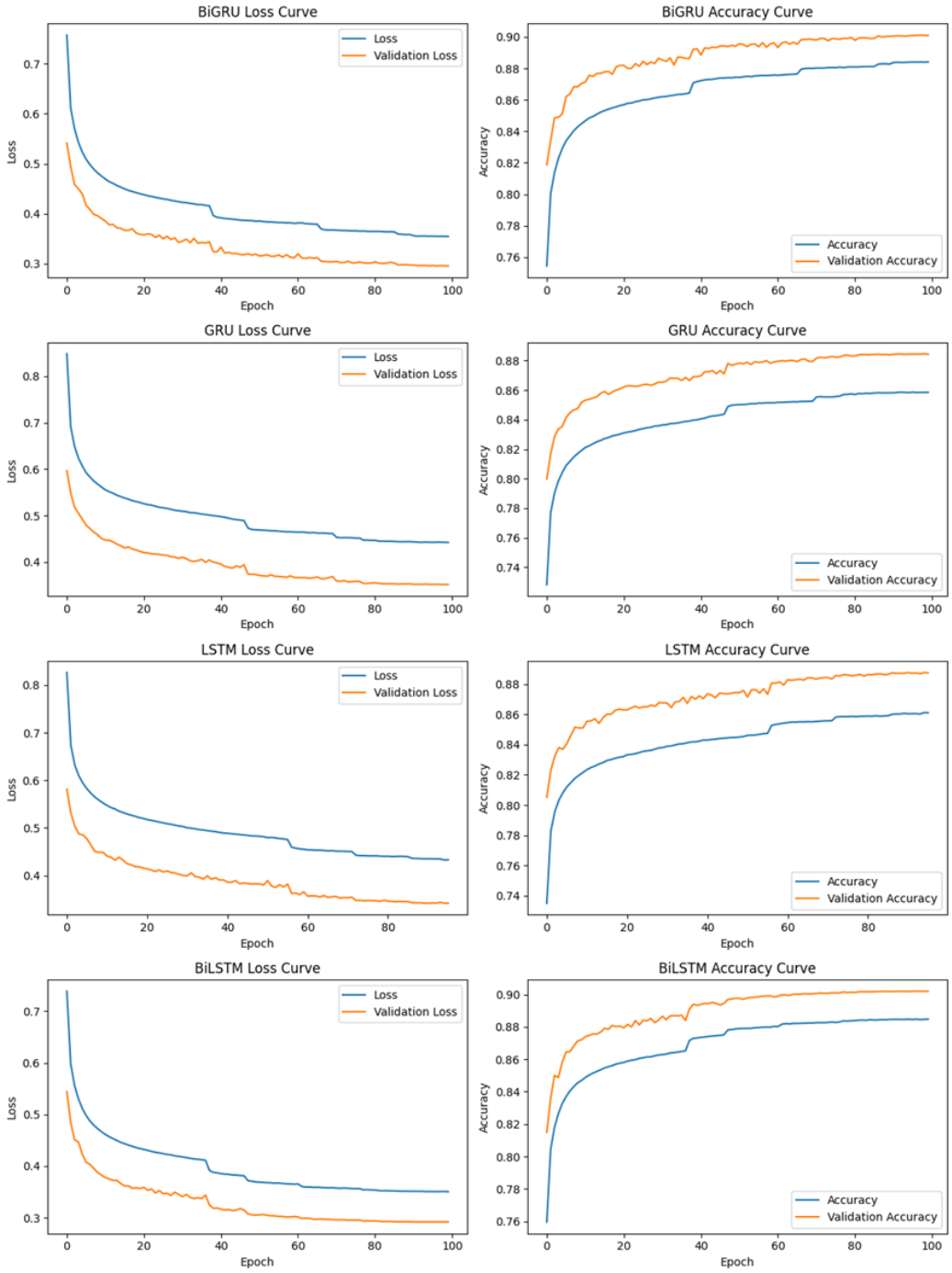


Figure 4. Individual application classification accuracy and loss curves.

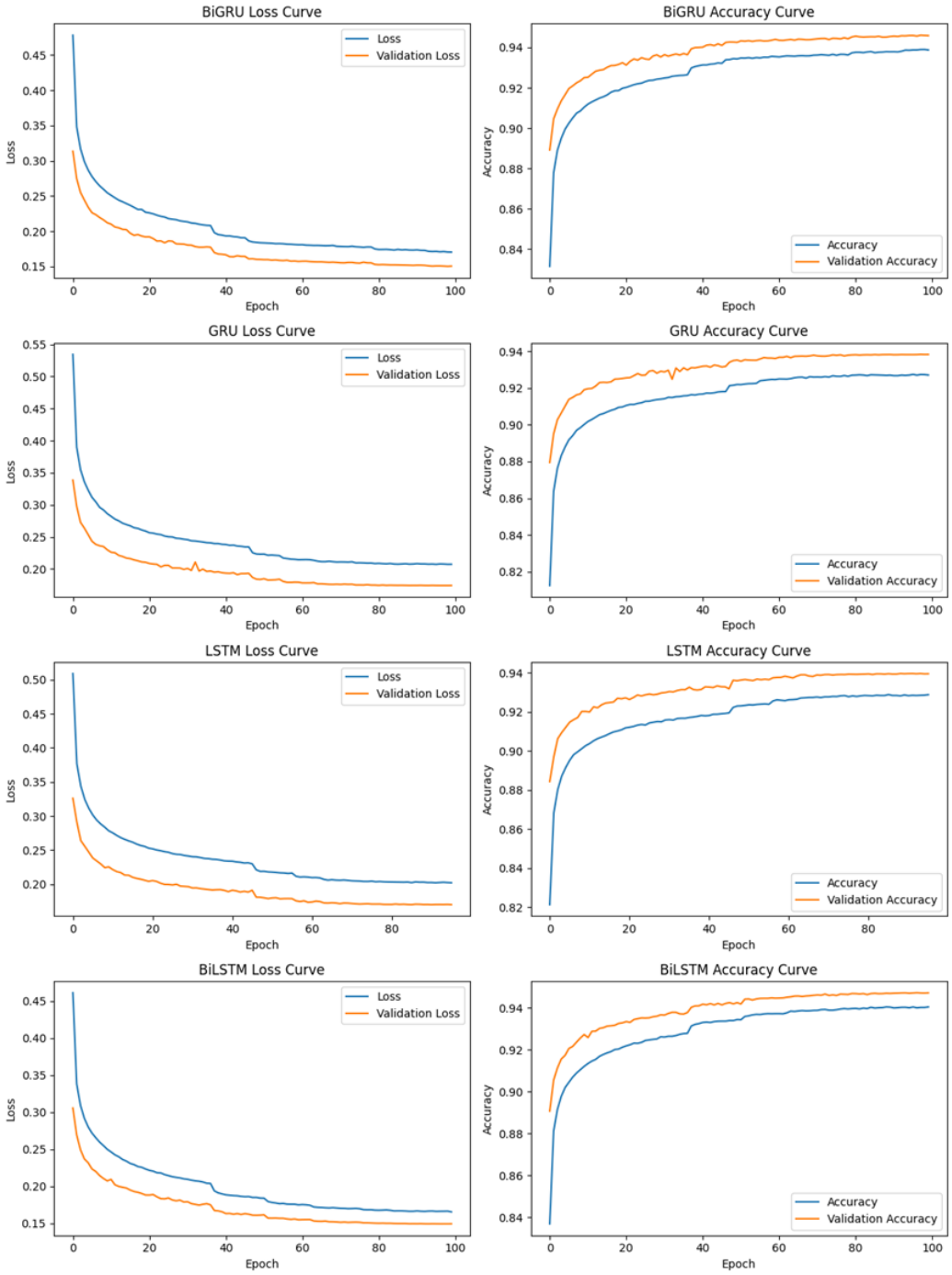


Figure 5. Application group classification accuracy and loss curves.

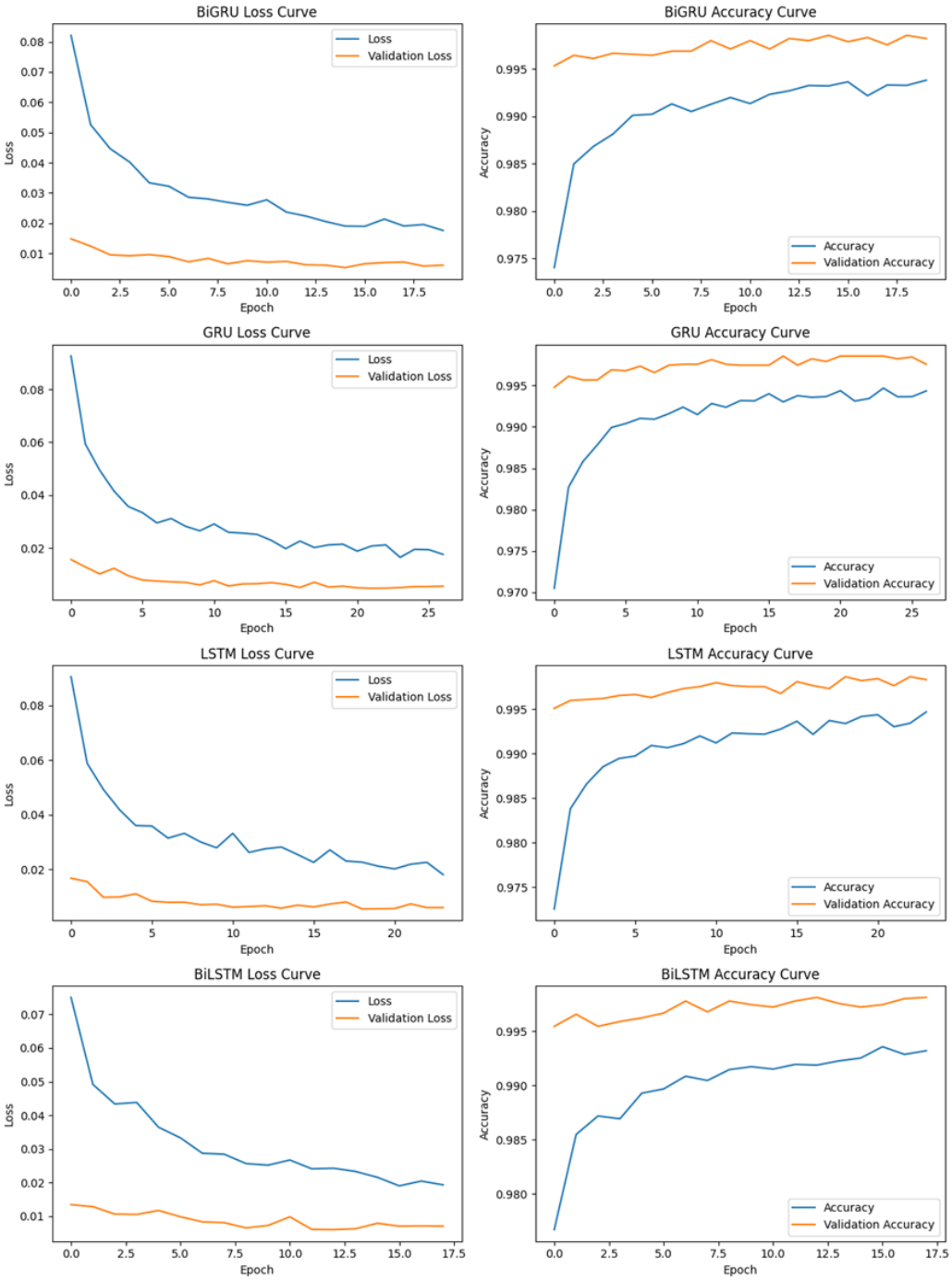


Figure 6. IoT logging and non-IoT classification accuracy and loss curves.

Table 6. Individual application classification confusion matrix (equal or above 5%).

Model	Application	TikTok	uTorrent	Spotify	Gmail
LSTM	BitTorrent		26.72, 17.02		
	Outlook				6.72, 7.92
	Facebook Messenger	5.26			
	Spotify	9.39			
	Snap Chat	9.61			
	X (twitter)	13.34			
BiLSTM	BitTorrent		18.67, 22.24		
	Outlook				5.38, 8.32
	Spotify	9.83			
	Snap Chat	9.40			
	X (twitter)	13.05			
	BitTorrent		16.65, 24.92		
	X (twitter)	13.48			
	Outlook			5.13	9.22,
	Spotify	9.41			
	Snap Chat	9.57			
	Facebook Messenger	5.79			
BiGRU	X (twitter)	11.96			
	BitTorrent		15.04, 24.10		
	Spotify	10.51			
	Snap Chat				
	Outlook	9.52		5.45	8.86
	Snap Chat	9.29			
	X (twitter)	12.79			

6. Discussions

The purpose of this study was to generate SDN flow-based dataset to address limitation of the current existing datasets to be used in SDN enabled campus networks. This section discusses how the developed dataset address the limitation stated in Section 2.

A central shortcoming of prior datasets is their collection in conventional network environments, which fail to capture the programmability, centralized control, and dynamic flow management inherent to SDN. In contrast, the dataset developed in this work was generated entirely within an SDN enabled testbed, which allowed precise manipulation of flow paths, device interactions, and traffic characteristics using a central controller. This contextual alignment between dataset preparation and target deployment environment was a key factor in the high classification performance achieved particularly in the binary IoT versus non-IoT scenario, where all models exceeded 99% accuracy.

Table 7. Individual application classification.

Application	Model															
	LSTM				BiLSTM				GRU				BiGRU			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
facebook	0.798891	0.780062	0.789364	0.885917	0.838402	0.810149	0.824033	0.901183	0.786688	0.775248	0.780926	0.88354	0.84232	0.80684	0.824198	0.90028
l2tpvpn	0.985797	0.99042	0.988103	0.885917	0.989958	0.993613	0.991782	0.901183	0.988509	0.987227	0.987868	0.88354	0.990541	0.992715	0.991627	0.90028
outlook	0.747938	0.826026	0.785045	0.885917	0.742992	0.867568	0.800462	0.901183	0.711555	0.853153	0.775947	0.88354	0.717579	0.881782	0.791251	0.90028
watersensor	0.973421	0.990329	0.981802	0.885917	0.980845	0.985394	0.983114	0.901183	0.976964	0.987763	0.982334	0.88354	0.983063	0.985197	0.984129	0.90028
dropbox	0.955441	0.955631	0.955536	0.885917	0.957363	0.969459	0.963373	0.901183	0.953213	0.954636	0.953924	0.88354	0.960463	0.966673	0.963558	0.90028
googlemeet	0.976847	0.930514	0.953118	0.885917	0.974042	0.93716	0.955245	0.901183	0.972298	0.933132	0.952312	0.88354	0.977722	0.936959	0.956906	0.90028
anydesk	0.985256	0.933117	0.958478	0.885917	0.985546	0.938278	0.961331	0.901183	0.982917	0.931499	0.956517	0.88354	0.975039	0.940706	0.957565	0.90028
spotify	0.774681	0.756477	0.765471	0.885917	0.826167	0.767925	0.795982	0.901183	0.772296	0.743523	0.757636	0.88354	0.818641	0.764712	0.790758	0.90028
filezilla	0.99708	0.995776	0.996428	0.885917	0.998287	0.99648	0.997383	0.901183	0.997681	0.994972	0.996324	0.88354	0.998188	0.996983	0.997585	0.90028
telegram	0.861678	0.940789	0.899498	0.885917	0.911826	0.952474	0.931707	0.901183	0.851895	0.940397	0.893961	0.88354	0.903403	0.951394	0.926778	0.90028
instagram	0.944727	0.889265	0.916158	0.885917	0.960585	0.903618	0.931231	0.901183	0.942409	0.887272	0.91401	0.88354	0.949979	0.906708	0.927839	0.90028
tiktok	0.534768	0.781859	0.635127	0.885917	0.556657	0.8264	0.665224	0.901183	0.525672	0.775974	0.626757	0.88354	0.565463	0.82295	0.670331	0.90028
zoom	0.977013	0.965173	0.971057	0.885917	0.99061	0.966904	0.978614	0.901183	0.972199	0.965071	0.968622	0.88354	0.986804	0.967108	0.976857	0.90028
googledrive	0.965949	0.941235	0.953432	0.885917	0.957785	0.952329	0.955049	0.901183	0.96907	0.936238	0.952371	0.88354	0.96578	0.95053	0.958094	0.90028
ssh	0.969517	0.979992	0.974726	0.885917	0.981188	0.986194	0.983685	0.901183	0.972776	0.97589	0.974331	0.88354	0.98009	0.984894	0.982486	0.90028
bittorrent	0.755211	0.661677	0.705357	0.885917	0.784498	0.717166	0.749322	0.901183	0.740617	0.665669	0.701146	0.88354	0.79061	0.694112	0.739225	0.90028
voip	0.98557	0.968108	0.976761	0.885917	0.990519	0.97317	0.981768	0.901183	0.983061	0.969525	0.976246	0.88354	0.9893	0.973575	0.981375	0.90028
gmail	0.761855	0.799779	0.780356	0.885917	0.809675	0.8214	0.815495	0.901183	0.814384	0.782281	0.79801	0.88354	0.814867	0.817981	0.816421	0.90028
uttorrent	0.685443	0.757459	0.719653	0.885917	0.740386	0.774804	0.757204	0.901183	0.689838	0.736743	0.712519	0.88354	0.732376	0.78878	0.759532	0.90028
netflix	0.984415	0.924432	0.953481	0.885917	0.983925	0.932397	0.957469	0.901183	0.985582	0.918757	0.950997	0.88354	0.985376	0.932497	0.958208	0.90028
openvpn	0.991795	0.994613	0.993202	0.885917	0.994232	0.99618	0.995205	0.901183	0.98928	0.994319	0.991793	0.88354	0.992585	0.996572	0.994575	0.90028
snapchat	0.811514	0.727062	0.76697	0.885917	0.829952	0.768043	0.797798	0.901183	0.808123	0.72908	0.766569	0.88354	0.838688	0.761482	0.798222	0.90028
temp-humidity	0.991119	0.972473	0.981707	0.885917	0.984699	0.981384	0.983039	0.901183	0.987392	0.977126	0.982233	0.88354	0.98493	0.983662	0.984295	0.90028
youtube	0.782738	0.845344	0.812837	0.885917	0.819511	0.853764	0.836287	0.901183	0.776828	0.847449	0.810604	0.88354	0.815754	0.851158	0.83308	0.90028
teamviewer	0.979335	0.935821	0.957084	0.885917	0.963434	0.941746	0.952467	0.901183	0.982063	0.929799	0.955216	0.88354	0.976897	0.939376	0.957769	0.90028

Application	Model															
	LSTM				BiLSTM				GRU				BiGRU			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
whatsapp	0.962025	0.890714	0.924997	0.885917	0.966748	0.908645	0.936796	0.901183	0.968681	0.879896	0.922156	0.88354	0.965881	0.907443	0.93575	0.90028
x-twitter	0.812297	0.778325	0.794948	0.885917	0.867331	0.787373	0.82542	0.901183	0.794552	0.774103	0.784194	0.88354	0.865632	0.78496	0.823325	0.90028
fbmessenger	0.856144	0.721562	0.783113	0.885917	0.867416	0.770998	0.81637	0.901183	0.862526	0.71557	0.782205	0.88354	0.860637	0.766004	0.810568	0.90028
ipsec	0.991729	0.997394	0.994554	0.885917	0.994105	0.997194	0.995647	0.901183	0.993704	0.996492	0.995096	0.88354	0.994499	0.996492	0.995495	0.90028
rdp	0.982764	0.940048	0.960932	0.885917	0.977933	0.952038	0.964812	0.901183	0.979316	0.941447	0.960008	0.88354	0.977748	0.952738	0.965081	0.90028

Table 8. Application group classification.

Application Group	Model															
	LSTM				BiLSTM				GRU				BiGRU			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
remote_connection	0.98145	0.938471	0.95948	0.93966	0.979143	0.945586	0.962072	0.94629	0.981912	0.935665	0.958231	0.93788	0.976087	0.944884	0.960232	0.94564
voip	0.986854	0.979918	0.983374	0.93966	0.987208	0.984135	0.985669	0.94629	0.987671	0.981323	0.984487	0.93788	0.988499	0.983834	0.986161	0.94564
iot_logging	0.997503	0.9987	0.998101	0.93966	0.997904	0.9997	0.998801	0.94629	0.997901	0.9987	0.998301	0.93788	0.997903	0.9996	0.998751	0.94564
video_conference	0.980182	0.949505	0.9646	0.93966	0.984101	0.953105	0.968355	0.94629	0.980151	0.948005	0.96381	0.93788	0.981501	0.954905	0.96802	0.94564
file_transfer	0.982836	0.970345	0.97655	0.93966	0.981399	0.977241	0.979316	0.94629	0.982218	0.96867	0.975397	0.93788	0.981854	0.975567	0.9787	0.94564
streaming	0.892428	0.816938	0.853016	0.93966	0.905285	0.834094	0.868232	0.94629	0.898223	0.812178	0.853036	0.93788	0.90503	0.82973	0.865746	0.94564
torrent	0.92978	0.950795	0.94017	0.93966	0.951217	0.953595	0.952405	0.94629	0.921987	0.950295	0.935927	0.93788	0.952909	0.953195	0.953052	0.94564
social_networking	0.813125	0.865558	0.838523	0.93966	0.831323	0.880462	0.855187	0.94629	0.80811	0.860793	0.83362	0.93788	0.830871	0.87813	0.853847	0.94564
vpn	0.988842	0.992302	0.990569	0.93966	0.991707	0.992202	0.991954	0.94629	0.988049	0.991802	0.989922	0.93788	0.992599	0.992102	0.99235	0.94564
mail_client	0.856945	0.933755	0.893703	0.93966	0.864959	0.942487	0.902061	0.94629	0.848285	0.931045	0.88774	0.93788	0.860816	0.944194	0.900579	0.94564

Table 9. IoT Logging and non-IoT classification.

Application Group	Model															
	LSTM				BiLSTM				GRU				BiGRU			
	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy	Precision	Recall	F1-Score	Accuracy
non_iot	0.998214	0.999007	0.998611	0.9986	0.998012	0.997021	0.997517	0.9975	0.997226	0.999206	0.998215	0.9982	0.998212	0.997816	0.998014	0.998
iot_logging	0.998992	0.998187	0.998589	0.9986	0.996981	0.997985	0.997483	0.9975	0.999193	0.99718	0.998185	0.9982	0.997785	0.998187	0.997986	0.998

Another recurring limitation of existing datasets discussed in this work was their narrow application coverage, which restricts a model's ability to operate in an environment with multiple applications, such as campus networks. The dataset presented in this study covers a broad spectrum of application categories, including social media, streaming, VPN, VoIP, IoT, torrent traffic, and cloud storage services. The mixture of different applications exposed models to a wider variety of traffic patterns, thereby making them more realistic to practical environments. While distinctive traffic types, such as VoIP, OpenVPN, and L2TP, achieved accuracy above 98%, applications with overlapping characteristics such as BitTorrent/uTorrent and TikTok/X exhibited higher misclassification rates. These outcomes highlight that broader coverage increases classification complexity but is essential for training models capable of performing reliably in heterogeneous SDN traffic environments.

A further limitation in many existing datasets lied in the anonymization of flow level information. This process can change traffic behavior and reduce the fidelity of model training. In this study, representative packet and flow level features were preserved while upholding privacy and ethical standards. This approach enabled models to learn from realistic patterns.

Finally, the integration of the SDN controller, multiple OVS bridges, and a live campus LAN introduced naturally occurring network dynamics, such as delay, congestion, and jitter. These natural conditions could be hard to obtain in a fully isolated laboratory dataset. These realistic timing variations enhanced the dataset's authenticity and improved the generalization capability of the trained models. By combining the experimental control of SDN with the variability of live network conditions, this work delivers a dataset that overcomes the principal shortcomings of previous efforts and provides a robust foundation for developing and evaluating DL models in SDN traffic classification.

Further insights from the evaluation of the datasets revealed that there is strong similarity in the network patterns generated by social network applications such as TikTok and X (formerly Twitter). This similarity rises primarily from the shared dependence on encrypted transport protocols, high frequency multimedia exchanges, and Content Delivery Networks (CDNs). Collectively, these approaches obscure application-specific flow characteristics. As a result, DL models may encounter high confusion rates when attempting to distinguish between these services as evidenced in the evaluation results of this study. To overcome this limitation, future classification models should incorporate mechanisms that go beyond static flow level features and instead leverage temporal dependencies, sequential packet behavior, and cross layer metadata. Additionally, hybrid approaches that combine deep sequence models (e.g., BiLSTM, GRU) with attention-based architectures may help in capturing subtle variations across traffic patterns. Such advancements would strengthen the robustness of classification systems against feature overlap and enable more accurate identification of social media applications within SDN environments.

7. Conclusion

This study highlighted the problem of application classification in SDN enabled campus networks related to the inadequacy of appropriate datasets for DL model training. To address the issue of the current dataset's lack of compatibility with the SDN environment, the study advocated the creation of a new dataset called SDNCampus. The study considered applications that represent the realistic usage patterns in SDN enabled campus networks. To improve smart networking, the study emphasized incorporation of IoT applications into

campus networks. Four (4) DL time series models were used to evaluate the dataset and demonstrated satisfying results.

Future work will consider enhancing the dataset's wide coverage by adding more applications especially related to operating systems traffic, adding more sensor data, and integrating application classifier for SDN enabled campus networks. The classifier will be used as an analytical tool to get insight into the application behavior of campus networks.

CONTRIBUTIONS OF CO-AUTHORS

Wilson C. Chanhemo	ORCID: 0000-0003-0218-7326	Conceived the idea and conducted experiments. Wrote initial version of the paper.
Mustafa H. Mohsini	ORCID: 0000-0003-0219-7946	Analyzed results and edited the paper.
Mohamedi M. Mjahidi	ORCID: 0000-0002-3962-6997	Provided technical information on methods and materials.
Florence U. Rashidi	ORCID: 0000-0001-6477-8362	Performed literature review, final editing and organized experiment resources.

SUPPLEMENTARY INFORMATION

The SDNCampus dataset can be found on Mendeley Data repository at https://data.mendeley.com/datasets/wvp9tksn72/1. The dataset is available in CSV file format for all 30 applications. Each file contains data for one application with 77 features. Files are named following the application that generated the flows to simplify identification.

REFERENCES

[1] Q. He, J. Lin, H. Fang, X. Wang, M. Huang, X. Yi, and K. Yu, "Integrating IoT and 6G: Applications of Edge Intelligence, Challenges, and Future Directions," *IEEE Trans. Serv. Comput.*, 2025, doi: 10.1109/TSC.2025.3586152.

[2] N. Makondo, H. I. Kobo, and T. E. Mathonsi, "The Latest Developments in Software Defined Networking: Adoption Rate and Challenges," in *Proc. IEEE AFRICON Conf.*, 2023, doi: 10.1109/AFRICON55910.2023.10293567.

[3] A. Finogeev, M. Deev, D. Parygin, and A. Finogeev, "Intelligent SDN Architecture with Fuzzy Neural Network and Blockchain for Monitoring Critical Events," *Appl. Artif. Intell.*, vol. **36**, no. 1, Dec. 2022, doi: 10.1080/08839514.2022.2145634.

[4] M. Tserenkhuu, M. D. Hossain, Y. Taenaka, and Y. Kadobayashi, "Intrusion Detection System Framework for SDN-Based IoT Networks Using Deep Learning Approaches With XAI-Based Feature Selection Techniques and Domain-Constrained Features," *IEEE Access*, Aug. 2025, doi: 10.1109/ACCESS.2025.3595595.

[5] M. H. Pathmaperuma, Y. Rahulamathavan, S. Dogan, and A. M. Kondoz, "Deep Learning for Encrypted Traffic Classification and Unknown Data Detection," *Sensors*, vol. **22**, no. 19, Oct. 2022, doi: 10.3390/s22197643.

[6] V. Tyagi and S. Singh, "Network Resource Management Mechanisms in SDN Enabled WSNs: A Comprehensive Review," *Comput. Sci. Rev.*, vol. **49**, Aug. 2023, doi: 10.1016/j.cosrev.2023.100569.

[7] D. Nuñez-Agurto, W. Fuertes, L. Marrone, E. Benavides-Astudillo, C. Coronel-Guerrero, and F. Perez, "A Novel Traffic Classification Approach by Employing Deep Learning on Software-Defined Networking," *Future Internet*, vol. **16**, no. 5, p. 153, 2024, doi: 10.3390/fi16050153.

[8] A. Tiwari, S. Kumar, and A. Rengarajan, "Incorporation of Deep Learning (DL) Techniques to Make Intelligent and Adaptable Wireless Communication Systems," in *Proc. Int. Conf. Adv. Comput., Commun. Inf. Technol. (ICAICCIT)*, 2023, pp. 1209–1213, doi: 10.1109/ICAICCIT60255.2023.10465798.

[9] T. Anagnostopoulos, P. Kostakos, A. Zaslavsky, I. Kantzavelou, N. Tsotsolas, I. Salmon, J. Morley, and R. Harle, "Challenges and Solutions of Surveillance Systems in IoT-Enabled Smart Campus: A Survey," *IEEE Access*, vol. **9**, pp. 131926–131954, 2021, doi: 10.1109/ACCESS.2021.3114447.

[10] J. Wang and Y. Wang, "Application of IoT and Cloud Computing Technologies in Smart Campus: Architecture Challenges and Solutions," in *Proc. 2nd Int. Conf. Internet Things Cloud Comput. Technol. (IoTCCT)*, 2024, pp. 146–152, doi: 10.1145/3702879.3702905.

[11] A. Azab, M. Khasawneh, S. Alrabaee, K. K. R. Choo, and M. Sarsour, "Network Traffic Classification: Techniques, Datasets, and Challenges," *Digit. Commun. Netw.*, vol. **10**, no. 3, pp. 676–692, 2024, doi: 10.1016/j.dcan.2022.09.009.

[12] A. Sharma and A. H. Lashkari, "A Survey on Encrypted Network Traffic: A Comprehensive Survey of Identification/Classification Techniques, Challenges, and Future Directions," *Comput. Netw.*, vol. **257**, p. 110984, Feb. 2025, doi: 10.1016/j.comnet.2024.110984.

[13] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Encrypted and VPN Traffic Using Time-Related Features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Priv. (ICISSP)*, 2016, pp. 407–414, doi: 10.5220/0005740704070414.

[14] A. Moore, D. Zuev, and M. Crogan, "Discriminators for Use in Flow-Based Classification," Dept. Comput. Sci., Queen Mary Univ. London, London, U.K., 2005.

[15] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of Tor Traffic Using Time Based Features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Priv. (ICISSP)*, 2017, pp. 253–262, doi: 10.5220/0006105602530262.

[16] MAWI Working Group, "MAWI Working Group Traffic Archive." [Online]. Available: https://mawi.wide.ad.jp/mawi/

[17] A. Maheshwari, B. Mehraj, M. S. Khan, and M. S. Idrisi, "An Optimized Weighted Voting Based Ensemble Model for DDoS Attack Detection and Mitigation in SDN Environment," *Microprocess. Microsyst.*, vol. **89**, Mar. 2022, doi: 10.1016/j.micpro.2021.104412.

[18] UNIBS, "Available Traces With Associated Ground Truth," Accessed: May 26, 2024. [Online]. Available: http://netweb.ing.unibs.it/ntw/tools/traces/

[19] L. Peng, H. Zhang, B. Yang, Y. Chen, and T. Wu, "Traffic Labeller: Collecting Internet Traffic Samples With Accurate Application Information," *China Commun.*, vol. **11**, no. 1, pp. 69–78, 2014, doi: 10.1109/CC.2014.6821309.

[20] G. Aceto, D. Ciuonzo, A. Montieri, V. Persico, and A. Pescapè, "MIRAGE: Mobile-App Traffic Capture and Ground-Truth Creation," in *Proc. 4th Int. Conf. Comput. Commun. Secur. (ICCCS)*, 2019, doi: 10.1109/CCCS.2019.8888137.

[21] K. Shahbar and A. N. Zincir-Heywood, "Anon17: Network Traffic Dataset of Anonymity Services," Tech. Rep. CS-2017-03, 2017.

[22] K. Jerabek, O. Rysavy, and I. Burgetova, "Measurement and Characterization of DNS Over HTTPS Traffic," *arXiv:2204.03975*, Apr. 2022. [Online]. Available: http://arxiv.org/abs/2204.03975

[23] M. Montazerishatoori, L. Davidson, G. Kaur, and A. H. Lashkari, "Detection of DoH Tunnels Using Time-Series Classification of Encrypted Traffic," in *Proc. IEEE Int. Conf. Dependable, Auton. Secure Comput. / Pervasive Intell. Comput. / Cloud Big Data Comput. / Cyber Sci. Technol. (DASC/PiCom/CBDCom/CyberSciTech)*, 2020, pp. 63–70, doi: 10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026.

[24] R. Wang, Z. Liu, Y. Cai, D. Tang, J. Yang, and Z. Yang, "Benchmark Data for Mobile App Traffic Research," in *Proc. ACM Int. Conf. Proc. Ser.*, 2018, pp. 402–411, doi: 10.1145/3286978.3287000.

[25] J. Luxemburk, K. Hynek, T. Čejka, A. Lukačovič, and P. Šiška, "CESNET-QUIC22: A Large One-Month QUIC Network Traffic Dataset From Backbone Lines," *Data Brief*, vol. **46**, p. 108888, Feb. 2023, doi: 10.1016/j.dib.2023.108888.

[26] S. Dadkhah, H. Mahdikhani, P. K. Danso, A. Zohourian, K. A. Truong, and A. A. Ghorbani, "Towards the Development of a Realistic Multidimensional IoT Profiling Dataset," in *Proc. 19th Annu. Int. Conf. Privacy, Secur. Trust (PST)*, 2022, doi: 10.1109/PST55820.2022.9851966.

[27] G. Wassie, J. Ding, and Y. Wondie, "Traffic Prediction in SDN for Explainable QoS Using Deep Learning Approach," *Sci. Rep.*, 2023, doi: 10.1038/s41598-023-46471-8.

[28] J. Luxemburk and T. Čejka, "Fine-Grained TLS Services Classification with Reject Option," *Comput. Netw.*, vol. 220, p. 109467, Jan. 2023, doi: 10.1016/j.comnet.2022.109467.

[29] J. Koumar, K. Hynek, J. Pešek, and T. Čejka, "NetTiSA: Extended IP Flow with Time-Series Features for Universal Bandwidth-Constrained High-Speed Network Traffic Classification," *Comput. Netw.*, vol. 240, p. 110147, Feb. 2024, doi: 10.1016/j.comnet.2023.110147.

[30] S. Jorgensen, J. Holodnak, J. Dempsey, K. De Souza, A. Raghunath, V. Rivet, N. Demoes, A. Alejos, and A. Wollaber, "Extensible Machine Learning for Encrypted Network Traffic Application Labeling via Uncertainty Quantification," *IEEE Trans. Artif. Intell.*, vol. **5**, no. 1, pp. 420–433, 2024, doi: 10.1109/TAI.2023.3244168.

[31] W. Shbair, T. Cholez, J. François, and I. Chrisment, "Open HTTPS Dataset," Accessed: Jun. 10, 2024. [Online]. Available: https://betternet.lhs.loria.fr/datasets/https/index.html

[32] I. Sharafaldin, A. Gharib, A. H. Lashkari, and A. A. Ghorbani, "Towards a Reliable Intrusion Detection Benchmark Dataset," *Softw. Netw.*, vol. **2017**, no. 1, pp. 177–200, 2017, doi: 10.13052/jsn2445-9739.2017.009.

[33] P. Velarde-Alvarado, H. Gonzalez, R. Martínez-Peláez, L. J. Mena, A. Ochoa-Brust, E. Moreno-García, V. G. Félix, and R. Ostos, "A Novel Framework for Generating Personalized Network Datasets for NIDS Based on Traffic Aggregation," *Sensors*, vol. **22**, no. 5, pp. 1–23, 2022, doi: 10.3390/s22051847.

[34] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection," *Comput. Secur.*, vol. **31**, no. 3, pp. 357–374, 2012, doi: 10.1016/j.cose.2011.12.012.

[35] A. Ferriyan, A. H. Thamrin, K. Takeda, and J. Murai, "Generating Network Intrusion Detection Dataset Based on Real and Encrypted Synthetic Attack Traffic," *Appl. Sci.*, vol. 11, no. 17, p. 7868, Aug. 2021, doi: 10.3390/app11177868.

[36] Canadian Institute for Cybersecurity, Univ. of New Brunswick, "CICFlowMeter (formerly ISCXFlowMeter)," Accessed: Jun. 12, 2024. [Online]. Available: https://www.unb.ca/cic/research/applications.html

[37] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. **23**, no. **13**, p. 5941, Jun. 2023, doi: 10.3390/s23135941.

[38] A. Rahmadeyan and Mustakim, "Long Short-Term Memory and Gated Recurrent Unit for Stock Price Prediction," *Procedia Comput. Sci.*, vol. **234**, pp. 204–212, 2024, doi: 10.1016/j.procs.2024.02.167.