Hindawi
Journal of Computer Networks and Communications
Volume 2023, Article ID 1495642, 18 pages
https://doi.org/10.1155/2023/1495642

Research Article

Elephant Flows Detection Using Deep Neural Network, Convolutional Neural Network, Long Short-Term Memory, and Autoencoder

Getahun Wassie Geremew [1] **and Jianguo Ding** [2]

[1] *IP Networking and Mobile Internet Track, Addis Ababa University, Addis Ababa, Ethiopia*
[2] *Department of Computer Science, Blekinge Institute of Technology, 37179 Karlskrona, Sweden*

Correspondence should be addressed to Getahun Wassie Geremew; getahunws12@gmail.com and Jianguo Ding; jianguo.ding@bth.se

Received 11 April 2022; Revised 23 November 2022; Accepted 8 April 2023; Published 22 June 2023

Academic Editor: Giovanni Pau

Currently, the widespread of real-time applications such as VoIP and videos-based applications require more data rates and reduced latency to ensure better quality of service (QoS). A well-designed traffic classification mechanism plays a major role for good QoS provision and network security verification. Port-based approaches and deep packet inspection (DPI) techniques have been used to classify and analyze network traffic flows. However, none of these methods can cope with the rapid growth of network traffic due to the increasing number of Internet users and the growth of real-time applications. As a result, these methods lead to network congestion, resulting in packet loss, delay, and inadequate QoS delivery. Recently, a deep learning approach has been explored to address the time-consumption and impracticality gaps of the abovementioned methods and maintain existing and future traffics of real-time applications. The aim of this research is then to design a dynamic traffic classifier that can detect elephant flows to prevent network congestion. Thus, we are motivated to provide efficient bandwidth and fast transmission requirements to many Internet users using SDN capability and the potential of deep learning. Specifically, DNN, CNN, LSTM, and Deep autoencoder are used to build elephant detection models that achieve an average accuracy of 99.12%, 98.17%, and 98.78%, respectively. Deep autoencoder is also one of the promising algorithms that do not require human class labeler. It achieves an accuracy of 97.95% with a loss of 0.13. Since the loss value is closer to zero, the performance of the model is good. Therefore, the study has a great importance to Internet service providers, Internet subscribers, as well as for future researchers in this area.

1. Introduction

Nowadays, Internet technology is turning to real-time applications that require high bit rates and strict delay for better quality of service (QoS) provision [1]. Real-time applications such as Voice over Internet Protocol (VoIP), video conferencing, online gaming [2, 3], online transactions [4], and virtual online classroom [5] become hot research areas for real-time applications due to the rapid growth of user interest in audio and video and the availability of integrated systems that can deliver multimedia data at a lower cost [6, 7]. These benefits have been achieved through the establishment of many multimedia institutions such as Google, Akamai, Level 3, Limelight, and Kankan [6, 8]. However, real-time applications demand a reliable and high-speed data (high startup and playback speed), characterized as large streams (elephant flows) or small streams (mouse flows), with stringent QoS and QoE requirements.

Elephant flows are less in number (10%) and exhibit long-lived flows that potentially fill network buffers end-to-end [9]. Elephant flows cause node congestion, delays, and packet loss if it is not managed appropriately [10]. To minimize or avoid this gap, our work aims to develop a dynamic traffic model to achieve a more predictable network and QoS in Software-Defined Networks (SDN) by using deep learning approach. The proposed model aims to

detect elephant flows to minimize network constraints such as latency, packet loss, and controller congestion. Several mechanisms have already been proposed to detect elephant flows [9]. However, these mechanisms do not provide a general and standardized method for elephant detection. Various fixed-sized thresholds have been explored to classify flows in open switches and controllers, which can lead to a high rate of false positives and false negatives [11]. Thus, there is a need to find standardized, dynamic, and optimal thresholds that can consider flow size, duration, packet size, and application type as heuristics, i.e., if these parameters are used as heuristics, the network is extensible and satisfies various QoS requirements. As IP networks combine the text, voice, and video data, dynamic flow classification is required to categorize traffic of different and future applications and provide the required services [6].

Traffic classification can be based on protocol types (e.g., UDP, TCP, FTP, or HTTP), application types (e.g., Skype, Chat, or Torrent), and traffic types (e.g., browsing, downloading, or video chat) [12, 13]. However, classifying traffic at application level becomes difficult because there are more than thousands of applications and new application are always being developed. Therefore, it is better to track the traffic flows at network layer by classifying as elephants and mice to achieve efficient QoS resource allocation. Therefore, developing a good classifier is one of the prerequisites for providing appropriate and adequate QoS and QoE in advanced traffic management of real-time applications [14]. The traffic classifier can be defined according to the classification objective, i.e., typical objective of this work is to provide efficient QoS and QoE provisioning [15].

Traffic classification is a key task for any Internet Service Providers (ISPs) or network administrators' QoS provision [16]. For this reason, it has been experimented using deep packet inception (DPI) [17, 18] and machine learning approaches [19]. To provide better Internet services according to application requirements, machine learning approaches are more recognized than DPI and of architectures' implementation [20]. Deep learning is the state of the art machine learning approach [21] that inspired us to develop our proposed stream classification model [21, 22]. Specifically, we developed an elephant flow detection model for SDN using deep neural network (DNN), convolutional neural network (CNN), long short-term memory (LSTM), and autoencoder algorithms. These algorithms have a dropout function to remove unnecessary information [23]. Moreover, they overcome the problem of overfitting and underfitting during model training by weight regularization (Adam), optimal epoch, and batch normalization techniques [24].

DNN is a multilayer perceptron (MLP) type of neural network that has input layer, hidden layer, and output layer in one forward direction without going backward transmission [25]. The enhanced NN and DNN are required to manage complex traffic from voice to video services. The quality of multimedia services on the Internet depends of congestions, failures, and other anomalies in the network. Therefore, we need a more advanced way to prevent these problems. DNN is a state-of-the-art neural network algorithm to develop a traffic management model that can guarantee resources for good QoS provision [18]. QoS measuring and resource prediction is possible with the DNN for distributed multimedia applications. The DNN can forecast future traffic variations as accurately as possible to predict the user network behavior.

The CNN is a feature extractor used to filter and preprocess data by defining local correlation between network neurons of neighboring layers to provide abstract representations of the input features. The loss function renders feedback signal for the learning purpose, and the optimizer is used to determine how learning proceeds [21, 26].

LSTM is an extension of the RNN that provides the capability of "long-term memory" in addition to short-term memory. It stores a list of all of the previous information in its memory and makes it available for training current neural network neuron [27, 28].

Deep autoencoder uses an input layer, bottleneck, and an output layer. It extracts features during training, and it usually works with the CNN. CNN-based autoencoder uses the CNN to filter content bearing words, numbers, or images values from the given input instance (record) [26].

The proposed classifier classifies traffics into elephant and mouse flows as the current traffic flow management is time-consuming and impractical to update the list of existing and future applications to transmit their heavy loaded traffic. Once we identify elephant flows, it is possible to implement a clustering model in SDN controller for route assignment of elephant flows based on elephant size to avoid overloading of links in a network.

In this work, research questions are prepared to answer the existing network QoS constraints. The questions include "By how much DNN, CNN, LSTM, and autoencoder detect elephant flows from mice flows for good QoS?." The performance of these algorithms is also seen in terms of accuracy and execution time over different QoS-based datasets. Optimal epoch iteration was experimented to overcome underfit and overfit challenges. The effect of batch size was also investigated during flow classification model development at constant and optimal epoch.

After presenting the introduction in Section 1, related work is discussed in Section 2. We presented our proposed traffic model architecture and modeling concepts in Section 3. In Section 4, experimentation results are discussed and evaluated. Finally, we presented the conclusion and future works in Section 5.

2. Related Work

The ever-growing number of applications with their huge and heterogeneous traffic needs more advanced traffic management mechanisms to achieve good end-to-end quality on the Internet [29]. In particular, real-time online applications (audio/video) require dynamic traffic management to prevent the negative effect of elephant flows in the network at runtime [3, 30].

Multimedia streaming, real-time interactive applications, and parallel processing in data centers are some of the examples of applications that require QoS guarantees for an

enhanced quality of experience (QoE) to users [2]. The need of good traffic management in real-time applications and QoS requirements has motivated many research works in the network field. The efforts resulted in several proposals with a number of research approaches. Research approaches that have been proposed for real-time traffic management include architecture-based [2, 31, 32], port-based [13, 33], payload-based [34], machine learning-based [10–12, 22, 30], and deep learning-based [21, 24, 35–39] approaches. However, the port-based and payload-based options are not efficient in detecting signatures from payloads after encryption [40]. Therefore, QoS cannot be achieved with these approaches. To achieve good QoS, authors such as Oliveira [2] focused on architecture-based solutions.

Oliveira [2] proposed an SDN-based architecture that provides QoS on distributed applications. The SDN constitutes an emerging paradigm that facilitates the creation and introduction of new abstractions in the network, simplifying management and facilitating traffic flows.

The SDN decouples network devices (switches and routers) as data plane and control planes. The control planes contain network intelligence (controller) to manage signal messages and data forwarding devices. SDN architecture provides a global view and an increased level of network programmability to create flexible capabilities and provide effective QoS provisioning mechanisms. The QoS provision architecture leverages SDN's class of service capabilities. It enables QoS requests and negotiations between applications and the SDN controller.

However, the solution negotiates small, medium, and large traffics with subscribers and service providers; traffic classification is not supported by the current dynamic traffic management mechanism.

In contrast to the architecture-based approach, machine-learning algorithms have recently gained more recognition due to their high performance in achieving good QoS requirements [41]. Machine-learning algorithms such as random forest, Naïve Bayes, neural network, and decision tree showed better performance in classifying traffic in data centers for deploying IOT and fog platforms for processing bills and other similar transactions [13, 30]. As the goal of machine learning is to identify sample data and build a learning model, it classifies the testing samples through the constructed classifier.

Many neural network algorithms such as recurrent neural networks of different types are used for classifying network traffic, focusing on feature selection, and extraction [30, 40]. Since feature and algorithm selections are very important for improving the classifier performance, SDN-based features such as actions, flow size, open flow protocol, duration, and application type are very important in addition to the usual five-tuple parameters such as source address, destination address, source port, destination port, and protocol.

Other authors also tried to mix machine learning and neural network algorithms to solve traffic identification problems. In this regard, Dong and Li [42] proposed a novel application identification method with multiple neural network layers to improve the efficiency and flexibility of application identification. A single application is treated in a single neural network module. Naïve Bayes algorithm was integrated within a single neural network module to classify traffic further from a single application.

Niloofar and Liu Bayat [43] identify the traffic flows of applications and services. Traffic classification involves extracting high level features from network packet data and then training with the CNN based on packet payload and interpacket arrival time parameters. The CNN was applied to online ML services, offline ML services to achieves users' QoS guarantee, and high system utilization. Based on a prediction model, a QoS-guided scheduling strategy can be proposed to identify the best placements for traffic.

Hamdan et al. [11] also conducted a study on traffic flow management in SDNs. An elephant detection technique was used to create two classifiers for SDN switches. In their work, most mouse flows in the switches can be filtered based on sketches statistics. Therefore, mouse flows cannot send requests and signaling messages to the controller to minimize the load on the controller.

Therefore, elephant flow detection becomes more interesting and dynamic when modeled with deep learning algorithms [35]. Deep learning is a branch of machine learning of neural networks [44], which has better learning ability for highly complex tasks than machine learning [36].

Ali [37] used deep learning algorithms including deep neural network in the SDN. In their paper, network traffic identification is an essential function for fine-grained traffic management task, although application classification based on application type cannot always detect elephant flows. Their study did not consider the detection of traffic from future fabricable applications, although the performance of the existing application classifier reaches 96% in terms of accuracy. The classifier was integrated with the controller, which overloads the controller. Instead, the flow classifier module can be integrated in OpenSwiches to share responsibility of the controller. Only elephant flows should be better reported to the controller for further elephant flow clusters. Moreover, the task of optimal route selection and assignment can be done by the controller as usual. Thus, the classification in the OpenSwitches and the clustering in the controller can work together to optimize QoS in the SDN. The classifier that improves QoS needs to be more dynamic and use deep learning algorithms. The state of the art is a deep learning algorithm such as deep neural network [37],CNN [43], LSTM [45], and deep autoencoder [39], which have been tested for traffic classification.

Lopez-Martin et al. [46] presented the potential of the CNN algorithm for classification tasks, as we intend to implement it for our elephant flow detection. The CNN is a classification algorithm that can initially be used automatically for representative filtering of traffic. By concatenating multiple CNNs, including dropout layer, maxpooling, and batch normalization layers, complex features can be extracted automatically. The dropout layer provides regularization (a generalization of results for unseen data) by omitting (setting to zero) a certain percentage of the output from the previous layer. This allows the network to not rely too heavily on a particular input, preventing overfitting and improving generalization. The maxpooling layer selects the maximum value of the traffic value

to reduce the number of features and the computational complexity of the network. The result is a representative output with less sampling. Batch normalization speeds up training and can improve performance results.

According to Ren-Hung [45], LSTM is used for classifying traffic from many modern software systems and applications for heterogeneous services in the data centers (clouds). The large growth in the number of these services has led to a critical criterion of QoS, which includes factors such as response time, location, and cost. As the value of dynamic QoS attributes vary with time, there is a need of advanced algorithms such as LSTM to accurately forecast future QoS values to identify routes or know a service may be about to fail in advance. Therefore, the LSTM-based neural network is very important to forecast future QoS values.

According to Franco et al. [47], an autoencoder can be classified into four types depending on the structure of the deep learning layer and regularization. These are vanilla autoencoder, denoising autoencoder, sparse autoencoder, and variational autoencoder. The one we are concerned about is the variational autoencoder (VAE). VAE is a deep generative model that can simultaneously learn a decoder and an encoder from data. An attractive feature of the VAE is that it estimates an implicit density model for a given dataset via the decoder. While learning a generative model for data, the decoder is the key object of interest. The encoder extracts useful features from dataset and learning a good representation. Learning good data representations before building the models is very important. This is the reason that deep learning, in particular VAE, solves the fundamental problems of machine learning algorithms to transfer to new training tasks.

The loss function of the autoencoder compares these predictions with the targets and produces a loss value during compression at the autoencoder bottleneck [16]. The result of the comparison is the loss value, which is a measure of how well the network's predictions match the expectations. The optimizer uses this loss value to update the weights of the network [26].

Convolutional neural networks (CNN) with variable autoencoders have shown remarkable classification performance. However, the CNN model is susceptible to noise and redundant information encapsulated in the high-dimensional raw input data, resulting in unstable and unreliable predictions. This problem can be solved by using autoencoders, which are unsupervised dimensionality reduction techniques that filter out noise and redundant information to produce robust and stable feature representations [48]. The experimental result showed that autoencoder-based binary classification enables to score an average precision of 97.49% after 10-fold cross-validation of elephant and mice flows [48]. Therefore, CNN-based AE can be one of the promising elephant flows detection algorithms in addition to pure supervised algorithms including DNN, CNN, LSTM, and other deep learning algorithms for the sake of QoS optimization.

A summary of previous works is presented in Table 1. The first group used deep learning (DL) techniques in line of traffic classification (TC), QoS-based datasets, real-time applications, and its state of the art deep learning algorithms. The second group used machine learning (ML).The third taxonomy used architecture-based solutions which employed network architecture for network management optimization such as software-defined network (SDN) architecture. The black circle in the last column indicates the focus of the work and the last row concerns our work in this paper.

3. Motivation

Accurate traffic classification is the basis for various network activities, including network traffic management and network security auditing [38]. Network traffic classification and analysis has been performed using port-based, DPI, and machine learning techniques. However, in recent years, the rapid increase in the number of Internet users and Internet traffic has led to network congestion. As a result, both the port-based and DPI approaches are becoming inefficient due to the exponential growth of Internet applications that incur high computational costs. The machine learning approach, especially the deep learning approach, has shown potential to detect traffic anomalies aligned with SDN traffic control capability. Therefore, we are motivated to develop a deep learning model for software-defined networks that can accurately distinguish elephant flows from mouse flows. The SDN and deep learning technologies are the state of the art traffic management techniques that we use to detect elephant flows for QoS optimization. This dynamic QoS optimization allows network administrators or ISP operators to dynamically predict traffic to prevent resource underutilization and network congestion due to resource overutilization [45, 51]. Given the current massive volume of traffic, ISPs need to predict the application type of a flow through the Internet in order to secure, monitor, and sufficiently allocate the QoS requirements of Internet users in advance [20].

(1) Intuitively, there are several reasons why network traffic classification can benefit Internet users, network administrators, and ISP operators.

(2) Developing a dynamic model in he SDN can minimize or avoid congestion problems, i.e., the Internet is constrained or unavailable due to inflexible traffic handling. This makes the network more flexible and programmable for network operators. It also ensures better QoS performance [34].

Integrating deep learning models into the SDN minimizes human intervention, i.e., it increases automation, and network administrators or operators can customize the network in terms of topology, configuration, and additional module integration in OpenSwitches and controllers. Thus, it opens doors for network administrators to manage the network in their context.

The abovementioned intuitions motivate us to explore network traffic classification using deep learning models in the SDN. The proposed elephant detection model encourages Internet subscribers to negotiate with service providers according to their QoS requirements.

Table 1: Summary of previous works.

Paper	Networking problem	Approach	Method/ classifier technique	Application type	Dataset used	Experimental result	SDN	ML/ DL	Elephant	Data center
Aceto et al. [30]	Encrypted TC	Deep learning	SAE, CNN, and LSTM	HTTP traffic	300 k Mobile datasets activity	SAE outperforms in TC	○	●	○	○
Nascita et al. [35]		Deep learning	1D-CNN	WeChat	MIRAGE 2019 dataset	Global model interpretation r	○	●	○	○
Ali[37]	Intelligence TC	Deep learning	Deep neural network	HTTP, mail, and multimedia	Moore dataset	Better accuracy, precision, recall, and FScore results	○	●	○	○
Dong and Xia [39]	Deep learning application	Deep learning	SC-CNN	Image segmentation	MNIST dataset	Deep learning create more powerful optimization methods	○	●	○	○
Dong et al. [38]	Abnormal traffic detection	Deep learning	Kmean, AE, and reinforcement	Abnormal traffic	NSL-KDD and AWID datasets	Achieved good result in time complexity	○	●	○	○
Dong et al. [36]	Tc	Deep learning	CNN and GAN	FTP, Gmail, and Skype	USTC-TFC2016 dataset	Yields better application traffic classification and detection result	○	●	○	●
Bovenzi et al. [29]	Model parallelism TC	Machine learning	RF, DT, and Bayes	IoT and fog platform	Anon17 NIMS dataset	Reducing training time	○	●	○	●
Chang [40]	Encrypted TC	Machine learning	RNN-AE	Github, Gmail, and Icloud	Real-world dataset 18 applications	Achieves 99.14% performance	○	●	○	○
Ibrahim [49]	Online game	Machine learning-mixed	Fixed Java code	http, FTP, and Skype	LOL game dataset	Produces 91% accuracy	○	●	○	○
Dong [20]	Multiclass TC	Machine learning	SVM	http, imap, and dns	MOORE and NOC datasets	Improve classification accuracy	○	●	○	○
Shi et al. [50]	Machine learning Application	Machine learning	Netflow extended machine learning	FCBF algorithm	Machine learning Application optimization	Algorithm called FCBF yields better performance	○	●	○	○
Dong and Li [42]	Traffic identification	Machine learning	Neural network and Naïve Bayes	TCP and UDP flows	MOORE and NOCSET	Achieves 95% identification accuracy	○	●	○	○
Shi [13]	Online encrypted	Machine learning	Naïve Bayes	Online Skype	Skype-SET	Reduces false positives and false negatives	○	●	○	○
Dong et al. [16]	Traffic identification	Algorithm based on architecture	High identification accuracy	Routing application	NOC_SET, CAIDA, and LBNL_SET	High accuracy	○	●	○	○
Oliveira [2]	QoS	Architecture	Fixed Python program	Distributed applications	Small text and video dataset	Low overhead	●	○	●	●
Hamdan et al. [11]	Load balancing	Architecture	Fixed Python program	d/t applications	Sketch-based filter elephants	Good running time and performance	●	○	●	●
This paper	Elephant flow detector	Deep learning	DNN, CNN, LSTM, and AE	Real-time apps	NIMS and SDN datasets	98.78% accuracy	●	●	●	●

4. Materials and Methods

This section presents dataset preparation and model development methods, and traffic classification model description and model evaluation techniques.

4.1. Dataset and Prepossessing. The training dataset is prepared from existing VoIP data, video streaming, and audio data transmission. The VoIP data comes from HTTP and GTalk applications from Network Information Management and Security Group (NIMS) dataset [52], which contains about 303,549 traffic flow records. We also used Unicauca-dataset, a network traffic dataset with 15,001 instances and 75 features. To evaluate the QoS provision in the SDN, the SDN dataset is used and tested by the selected deep learning algorithms.

The three datasets are modified assuming QoS and stored in a CSV file. The datasets are used to improve the QoS requirements of applications with elephant and mouse flow classes [53]. Classification was based on packet size, duration, flow size, byte count, and application type as heuristics. We added a class parameter that included elephant flow (1) and mouse flow (0) classes as the last column. Specific heuristics we used in categorizing mouse and

elephant flows are duration, packet size or flow size, and number of bytes in a flow. Mouse flows take at least 10 seconds on average [54]. Each short flow requires less than approximately 15 packets [55] and each packet contains 500 bytes [15]. The data preprocessing is accomplished based on parameterization of real-time and non-real-time applications. During and after data processing, packet sampling issue has been widely done [39]. Accordingly, we used stratified 10-fold cross-validation [12] to evaluate the performance of the models with new unseen traffic predictions. The 10-fold cross-validation method is used commonly for precision with the premise that a traffic dataset is divided into 10 parts, 9 of which constitute the training data with 1 representing the test data [13].

4.1.1. Feature Selection and Extraction. It is important to take into account the impact of traffic features and application categories for the development of classification models. Packet payload information becomes a big obstacle to identify QoS-based traffic flows. Instead, we have to see network traffic in line of flows by correlating many features and application categories. When feature correlation is 0, it represents that two variables are independent, if feature correlation is 1, it shows two variables have a strictly functional relationship [50]. Combining two or more feature characteristics such as packet size, traffic duration, and types of applications can yield better flow classification accuracy.

Network traffic is represented by flow-based features. We used the feature prepared for NIMS dataset as an example [52] and added an additional column for elephant and mouse flow classes. We used the packet size threshold used by Cisco systems in data centers to determine dynamic thresholds [55]. Cisco referred to as an elephant flow if the flow contains more than 15 packet sizes, i.e., short flow is less than 15 packets. We also consider byte size to refer to the flow as elephant or mice. Packet sizes are typically greater or equal to 500 bytes per packet [55]. Mouse flows have a size of 10 KB in OpenSwitches of datacenters [15, 54] and an average duration of 10 s [54].

As can be seen in Table 2, the last row describes elephant and mice categories. Network flow is described by a set of statistical features which can be calculated from one or more packets of flows and compute feature values [52].

4.2. Deep Learning Techniques. Deep learning techniques became popular due to the explosive growth and availability of data (big data) and the increase of high-performance computing hardware such as graphics processing unit (GPU) to train large amounts of data [56]. It takes longer training time to yield higher accuracy due to its ability to process a large number of features [57].

Deep learning algorithm passes the data through several training batches and layers to yield complex correlation (models) between features [57].

Deep learning models have been recently studied for network traffic classification learnings. At present, the deep learning techniques include deep neural network (DNN), convolution neural network (CNN), long short-term memory (LSTM), deep autoencoder, deep Boltzmann machine, generative adversarial networks, and so on [21].

4.3. Proposed Traffic Classification Models Description. Network traffic classification is a key component for network management and QoS management. Therefore, employing deep learning methods can distinguish network traffics from distributed and multimedia applications [58]. The proposed traffic classification models enable us to obtain better classification results and reduce the classification time through overall optimization without excessive manual intervention, especially in the deep autoencoder model [59].

The proposed traffic classification models are developed with deep learning algorithms. Deep learning algorithms, including DNN, CNN, LSTM, and autoencoder, use loss function and optimizer components to build and evaluate the elephant detection model. The elephant detection model training continues until the final classifier is built as per epoch (upto 50) and batch size settings (128). The final classifier is obtained after many updates of weights.

The selected algorithms usually yield better classification performance than other general machine learning algorithms [52]. Therefore, we used these deep learning algorithms to categorize traffic flows into elephant and mouse flows based on tangible attributes including flow size, flow size, total packet size, protocol type, application type, and flow duration as heuristics information for QoS provision. Elephant flows are flows that take long time and have large packet size, whereas mouse flows are those with relatively low sizes transmitting for a short duration [15]. Deep learning categorizes not only elephant flows and mouse flows but also helps to further cluster elephant flows for more manageable traffic flows. Proper deployment of a traffic load analysis provides valuable insights, including how busy a link is, the average delays, and the average packet size for wise use of path resources. Thus, deep learning based traffic classification model have advantages in minimizing time complexity and achieved good results on top of QoS datasets [38]. The selected deep learning algorithms for building the proposed traffic classification model are discussed. In particular, the deep Autoencoder is more described and demonstrated diagrammatically since it is one of the cost-effective methods due to its automatic training capability in unsupervised manner without human intervention.

4.3.1. DNN. The DNN is typically feed forward network type of multilayer perceptron (MLP) in which data flow from the input layer to the output layer in one forward direction without going backward [25]. Training a neural network comprises sequence of layers, which are combined into a network having the input data (traffic flow) and corresponding expected targets (elephant or mice). Some formulated neural network models used to identify real-time application layer protocol, and the work yields lower time and space complexity [42].

These DNN components create chains and map the input data to predictions. The layers layer 1, layer 2, and layer

Table 2: Attributes of NIMS dataset.

Attributes	Duration and size of the flow
Min forward interarrival time	Min backward interarrival time
Std. deviation of forward interarrival time	Std. deviation of backward interarrival time
Mean forward interarrival time	Mean backward interarrival time
Max forward interarrival time	Max backward interarrival time
Min forward packet length	Min backward packet length
Max forward packet length	Max backward packet length
Std. deviation of forward packet length	Std. deviation of backward packet length
Mean backward packet length	Mean forward packet length
Duration	The time taken for arrival of packets
Protocol	Application layer protocol (HTTP and GTalk)
Total_fpackets	Total number of bytes in forward direction
Total_fvolume	Total volume of bytes in forward direction
Total_bpackets	Total number of bytes in backward direction
Total_bvolume	Total number of bytes in forward direction Total volume of bytes backward direction
Class	Elephant and mice labels

n are fundamental data structure which is the building blocks of deep learning model formulation. Models are networks of layers that can be a formula or algorithm [26]. Layers can be input, hidden (dense), and output. Each layer can have different number of neurons and it can be calculated using parameters, input (i), weight (w) and bias (b), and output (y) as it is formulated in the following equation:

$$y = iw + b. \tag{1}$$

Activation function enables the DL model to learn complex patterns. The most frequently used activation function is ReLU which we use it for our traffic classification model [60].

The gap between the actual output (y) and expected output (y') is recorded as loss. The loss function compares the prediction values to the target values to produce a loss value. The model is the classifier (elephant flow detector) which predicts the unseen flow to their target category as per the assignment of class by data expert manually (true target).

The loss value shows us how well the network's predictions match what was expected. If the loss value is high, the optimizer updates the weights to minimize the loss value. The loss function sets feedback signal for learning purpose. We measure not only the loss incurred during training, but also the performance of the model in terms of accuracy.

The optimizer is used to determine how learning proceeds by letting weight update.

4.3.2. CNN. The feed forward neural network enables end-to-end training from input to out layer by exploiting existing deep learning technology [61]. The CNN requires an input filtering function and data preprocessing mechanism by defining local correlation between network neurons of neighboring layers to deliver abstract representations of the input features [26]. It is one of the deep learning algorithms which have 1D, 2D, and 3D maxpooling filters to reduce the network scale and further reduce the computation load on the pooling filter. We used 1D traffic data, and local features are combined to form the global features; then, the pooling filter is used to remove the unnecessary information to obtain abstract data of reduced size [23].

So, we can drive that the CNN is a special MLP which we employee for elephant flow detection. A normal CNN model consists of different types of layers which allow the model to learn and extract features relevant to classes [62].

(1) Convolution Layer. This is the layer where n number of filters is applied to extract features based on the given size of the kernel.

Batch Normalization. Batch normalization is used to normalize the output of one convolution going as an input to another convolution. This results in efficient training and helps in reducing overfitting [63].

(2) Max Pooling. Max pooling layer is used to reduce the dimensionality of the feature map by selecting the maximum value of a particular region based on the kernel size.

(3) Dropout Layer. This layer is used to reduce overfitting by dropping the specified percentage of features from the model. If dropout (noise) is being applied at training time, random changes in the model can happen at training time and are working to prevent overfitting during training. If this is not being applied at validation time, then validation accuracy could be higher than training accuracy [63].

(4) Fully Connected Layer. This is the final dense layer that is mostly used at the end of the network for classification. Unlike pooling and convolution, it has a global operation capability. It takes input from feature extraction stages, globally analyses the output of all the preceding layers and classifies the traffic data as elephant (1) or mice (0) [25].

4.3.3. LSTM. The recurrent neural network (RNN) provides the capability of "short-term memory" which allowed the use of the previous information at a certain point only to be

used for the present training task. The LSTM is an extension of the RNN which provides the capability of "long-term memory" where a list of all of the previous information is available for training the current neural network neuron [27]. LSTM's main components are the memory cells and the input, forgets, and output gates. These components allow the LSTM network to have connections from previous time steps and layers, where every output is influenced by the input as well as the historical inputs [28].

The LSTM uses feedback loops which lets weight update for correct class assignment during model training [27]. So, in this work, we also employ LSTM for the traffic classification task. Elephant flows can be detected using multiple LSTM layers, where each layer comprises many LSTM units, and each unit comprises input, forget, and output gates. To prevent overfitting problem, we used weight regularization (Adam), dropout, optimal epoch, and appropriate batch normalization techniques [24].

4.3.4. Autoencoder (AE). AE is categorized under unsupervised learning algorithm. It is a type of neural network that does not require the human labeling of data.

It reconstructs the input to an output of fewer dimensions [64]. The AE is trained to reconstruct its input through a bottleneck layer with fewer dimensions than the data space. The input (training data) and network maps together have different layers including input, encoders, bottleneck, decoders, and output. The AE first encodes the input to a hidden representation (code) of lower dimensions and then decodes it back into a reconstruction.

Even though AE is a clustering algorithm, it can be used for traffic classification by learning dynamic threshold value. The threshold value is obtained dynamically from the model learning process. The threshold value identification is the first task of AE during classifier formulation. So, class labels are not necessary for elephant and mice classification, rather class labels are dropped from the datasets unlike the training way of DNN, CNN, and LSTM. Only threshold criterion is used to detect elephant flows to optimize QoS provision [65, 66].

Elephant flow detection is accomplished if the traffic size and duration is greater than the threshold value. For example, if the threshold value is greater than 0.5, the traffic flow will be assigned in the elephant flow category unless otherwise it is a mice flow.

As AE is depicted in Figure 1, it takes network traffic data then filters the representative neurons from each instances then, autoencoder continues compressing the filtered data to find the threshold value.

Traffic inputs (T) are given to the autoencoder network, then it encodes to a coded form with minimal features at the bottleneck layer. The coded feature is decoded to yield the output layer (O). Threshold value is calculated during reconstruction of the network. We use the coded value (threshold) for elephant and mice flow classification task. In Figure 1, Unicauca-dataset is used to design the autoencoder structure. Unicauca-dataset has 87 attributes and we add one additional class column. Total, there are 88 parameters used as input starting from T1 to T88.

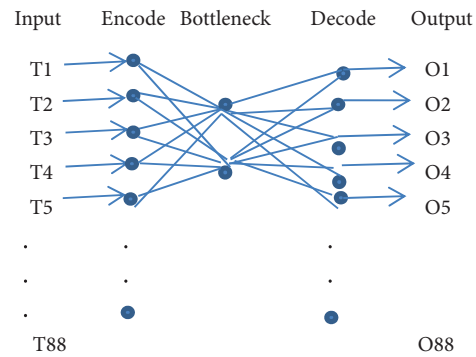

Figure 1: Autoencoder for network traffic data.

As visualized in Figure 1, we can take an unlabeled dataset and give to autoencoder for learning task. Let us take the original input T as (T1, T2, T3, T4, T5, . . ., T88), and the output, O as (O1, O2, O3, O4, O5, . . ., O88), is a reconstruction of the output. Autoencoder can be trained by minimizing the reconstruction error, e (T, O), which measures the differences between our original input and the consequent reconstruction. This construction loss yields a threshold value for classification task.

Thus, an autoencoder approximates an identity of the traffic whether the traffic instance is elephant or mice based on the threshold value, i.e., the hidden layer has two outlets, one for elephant flow and the other is for mice flows [37].

The traffic classification model workflow is also presented in Figure 2 having input traffic data (elephant and mice).

Firstly, elephant and mice traffic flows are inputted to the workflow. The AE model is then formulated to provide a threshold value which is used as the boundary during elephant flow detection, i.e., the AE model yields the threshhold value which is a formula to identify elephants in a nonlinear hyperplane manner.

Once the formula (hyperplane boundary) is found, traffic prediction is performed based on the threshold value. If the traffic input-weight size is greater than this threshold boundary, the flow is detected as elephant as it depicted in Figure 2. Unless otherwise; it is categorized under the mice class. So, the network setting does not change because the traffic does not lead to network congestion.

5. Experimentation and Evaluation

The Internet traffic measurement and analysis minimizes or avoids traffic congestion challenges [49], but prior to data transmission, the performance of the classifier must be verified with various metrics. We present the results of the four deep learning models on three dataset that we have adapted. We also compare the performances of the models. To design a robust evaluation framework, we run every model over 20 to 50 epochs. The average performance of each algorithm is considered as the promising potential of the elephant detection model. We use accuracy and loss metrics for model evaluation. We also checked the effects of overfitting and underfitting of models by conducting experiments on epoch repetitions and training data batch sizes.

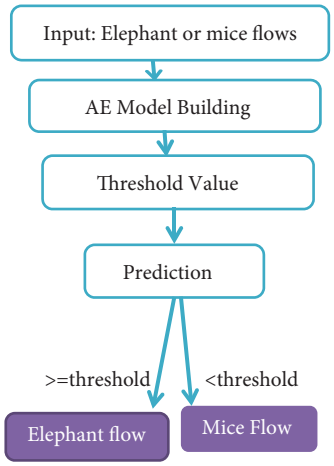

Figure 2: Model prediction using threshold.

5.1. Dataset. The goal of this study is to develop a model that is best suited to detect elephant flows. To achieve this goal, we selected an inelastic and an elastic application-based dataset. Thus, we used features from the NIMS dataset and took a non-real-time HTTP application as an example and Gtalk as a real-time application to formulate the model [52]. We then changed the class column as elephant flow (1) and mouse flow (0). We also used two other SDN-based datasets to test the performance of the proposed model in OpenSwiches. The datasets were divided into training, validation, and testing sets with 10-fold stratified cross-validation.

The training parameters, epochs, batch size, learning rate, and optimizers used in this model are listed in Table 3.

5.2. Experimental Setups and Tools Used. We installed Anaconda version 3 on an Intel (R) Core (TM) i7-4500U CPU @ 1.80 GHz 2.40 GHz laptop computer. We also installed components to help us perform the deep learning experiments. Some of the components are Jumpy, Pandas, and Matplotlib to process a variety of data and graph the experimental results. To control traffic and support QoS, we group incoming traffic into elephant and mouse flows.

The time required to complete one round of execution of each model is recorded in seconds. The models are built using the Python version 3.9 programming language and the Tensor Flow 2.3.0 and Keras 2.4.5 frameworks.

In deep learning, the input data are trained by automatically learning structured feature representations using the Keras framework [37]. Keras is a powerful and easy-to-use Python library for developing and evaluating deep learning models based on Tensorflow. Tenssorflow enables the definition and training of network models as a multi-dimensional array or list [26, 67]. Keras and Tensorflow modules were installed on Anaconda to obtain deep learning based autoencoder libraries. We implement DNN, CNN, LSTM, and autoencoder algorithms on SDN datasets. We run CNN and LSTM deep learning codes on Googlecolab to obtain fast computing performance.

Table 3: Training parameters.

Model	Epoch	Batch	Rate	Optimizer
DNN CNN SAE LSTM	20/50	128/512	0.01	Adam

5.3. Model Evaluation. The proposed model is evaluated using 10-fold stratified cross-validation on a test set. A classification result has four cases: true positive (TP), false positive (FP), true negative (TN), and false negative (FN). For our purpose, we used the same input form, the same training set, the same learning rate, and the same optimizer.

For cross-validation, we stratified the SDN dataset into 10-fold. The SDN dataset consists of a large number of instances. The dataset is automatically split into a training set and a test set. The training and test datasets are partitioned in a stratified manner, starting with fold 1 and ending with fold 10, as shown in Table 4.

To measure the performance, the metrics used is accuracy which is defined as follows:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{FP} + \text{TN} + \text{FN})}. \quad (2)$$

5.4. Discussion of Experimental Results. In this experimentation, we use DNN, CNN, LSTM, and autoencoder algorithms for our traffic classification. The experiment results are presented for each research; deep learning algorithms used are discussed. We used experimental results of the training history to measure performance of the models in terms of accuracy and loss for each employed algorithms.

Experiment 1. To what extent do DNN, CNN, LSTM, and autoencoder neural network algorithms detect elephant flows from mouse flows for good QoS?

5.4.1. Experimenting Elephant Flow Detection Using DNN. Network's elephant detection model classifies elephants and mouse categories to the target outputs. When we see the performance of the model in terms of accuracy using Adam optimization, training accuracy increases and training accuracy reaches 99.99% under epoch 50 on the NIMS dataset. The validation accuracy reaches 100% under the same epoch within 3 s and 5 ms.

The model training history of the model and performance accuracy of the DNN algorithm are shown in Figure 3(a). It shows the training accuracy and validation accuracy of the model. The final traffic classifier model is then verified with the best (i.e., highest) validation accuracy [68].

Figure 3(b) shows the training loss and validation loss of the model. The final model is the checkpoint with the lowest validation loss, 0.0037 which is closer to zero [68].

Table 4: Stratified 10-fold on SDN dataset for cross-validation.

Fold: 1, training set: 93455, test set: 10384
Fold: 2, training set: 93455, test set: 10384
Fold: 3, training set: 93455, test set: 10384
Fold: 4, training set: 93455, test set: 10384
Fold: 5, training set: 93455, test set: 10384
Fold: 6, trainng set: 93455, test set: 10384
Fold: 7, training set: 93455, test set: 10384
Fold: 8, training set: 93455, test set: 10384
Fold: 9, training set: 93455, test set: 10384
Fold: 10, training set: 93456, test set: 10383

The training loss diminishes starting from 0.0793 to 0.0022 as it is seen in (b) above. The test validation of the model using 10-fold stratified cross-validation is presented in the confusion matrix from Figure 4.

False positive and false negative errors are reduced to zero by the judicious use of heuristics that take into account the duration of data flow, the size of data flow, the type of application, and the size of the packets.

We also verified the performance of traffic flow classifier with a labeled application dataset, Unicauca-dataset. The total dataset has 15, 001 instances and 87 parameters. We modified the dataset considering the QoS requirements of applications and stored in a CSV file. We add the 88th column as class column that holds elephant flow (1) and mouse flow (0) classes. The duration of the mouse flow requires at least 10 seconds. Each flow has at least 15 packets, and each packet contains 500 bytes as it is stated in our methods. Having this heuristic information, we create a learning model that can detect elephant flows. The model performs 97.36% training accuracy within 7 ms and 97.24% testing accuracy using the deep neural network on Unicauca-dataset [69] as it is seen in Figure 5(a). The validation accuracy was constant while the training accuracy increases radically starting from 70.03% to 96.52%.

The loss that occurred during training on Unicauca-dataset showed radical reduction starting from 0.6443 to 0.1199, as shown in Figure 5(b). As training loss is presented, the training loss scored approaches to zero as the final model is the checkpoint with the lowest validation loss.

The validation of the model is demonstrated by he confusion matrix in Figure 6.

The main objective of this work is traffic classification on the SDN dataset using state of the art DNN, CNN, LSTM, and autoencoder algorithms. The dataset was generated from RYU controller. It has 104,346 instances with 23 features. We modified the last column and substituted by elephant (1) and mice (0) classes after identifying duration, packet size, byte size, flow size, and application protocols parameter as heuristics.

The DNN experimental result obtained within 50 epochs is 99.97% training accuracy and validation accuracy is 100% in 1 s and 5 ms as it is seen in Figure 7(a).

We checked the performance of the classifier by performing 10-fold stratified cross-validation with the test set during evaluation. We achieved a test accuracy of 100%, which is promising model performance [70], interpreted to mean that test accuracy should not be higher than training accuracy. When the model is not overfitted, the training model classify's new samples as the training model is optimized for the testing samples latter [63].

The training loss decreases from 0.2587 to 0.0014 which is generally close to zero, as shown in Figure 7(b). 0.0014 noise is a normal occurrence, as we do not expect 100% perfection from machines, as humans have natural limitation to perform 100% autonomously. Humans detect during the day and night with accuracy of 80% average. The overall accuracy is usually expressed as a percent, with 100% accuracy is being a perfect model; achieving 100% recognition is a very difficult task for machines [71].

Let us explain our model performance using the confusion matrix. The prediction classes of the models are predicted values and actual values along with the total number of predictions [20]. Predicted values are those values, which are predicted by the model, and actual values are the true values for the given observations labeled by us, as shown in Figure 8.

True negative: 192 records were assigned to mouse streams as we annotated them as mouse flow.

True positive: We predicted elephant flows were positive (1) and 10187 records are actually truly mapped to elephant flows in data centers.

False negative: The model predicted elephant and mouse flow without any error.

False positive: The model has predicted 4 mouse records as elephants mistakenly. Thus, the error is only 0.04%.

(1) Discussion on Performance Result of DNN Algorithm. Model accuracy and explanations are relevant for many practical applications of deep neural networks, such as our traffic classification task. Accordingly, we found elephant classification model to be 97.36%, 99.99%, and 99.97% on NIMS, Unicauca, and SDN datasets, respectively. When we calculate the training accuracy, the average performance of the DNN is 99.12% for the elephant detection model. The model performance implies that our elephant detecting model can lead to better generalization performance under the DNN [72]. The DNN optimizes classification tasks with respect to training data in a heterogeneous manner as we tested on heterogeneous traffic including datasets generated from legacy and SDN networks. With sufficient parametric flexibility, these types of models can fit generalizable features and memorize nongeneralizable features concurrently during training [73]. The built model meets the general standard of deep learning models, i.e., the validation and test accuracies are greater than 96% and all validation and test accuracies were less than the training accuracy [63].

We can infer from the graphs Figures 3(b), 5(b), and 7(b), both training and validation loss are decreasing further, which is generally near to zero. Little noise is a normal occurrence since we do not expect 100% perfection from machines as human beings cannot perform 100% autonomously [71].

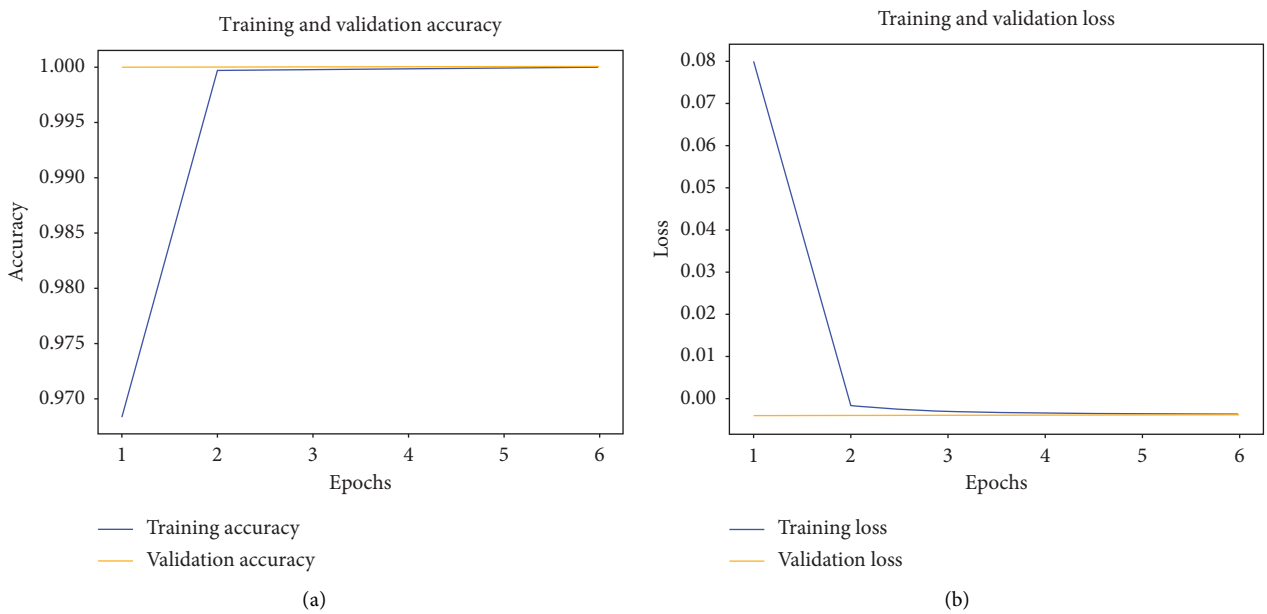

Figure 3: DNN performance result using NIMS dataset: (a) training accuracy and validation accuracy and (b) training loss and validation loss.

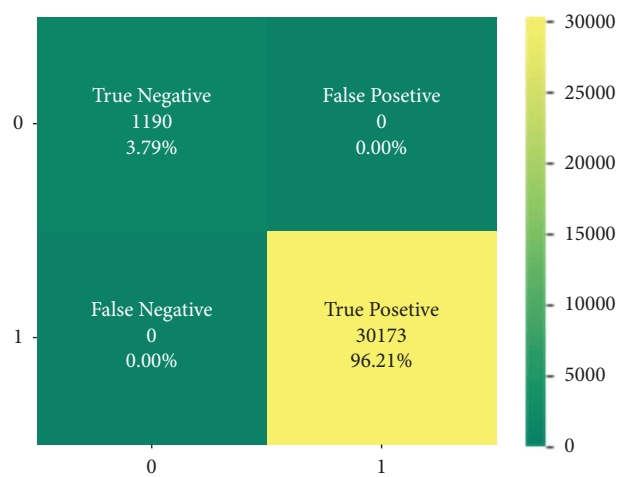

Figure 4: Confusion matrix on NIMS dataset.

(2) Experimenting Elephant Flow Detection Using Autoencoder. The goal of the autoencoder is to find optimal model parameters for the minimization of a loss function. The msle loss function on training samples finds the maximum loss value [74]. For this work, mean squared error loss function (msle) was used with the Adam optimizer and the loss approaches zero as it is seen in Figure 9.

Distribution of the deep autoencoder model, reconstruction loss distribution visualization, and the loss decreases uniformly and approaches zero over training history. In particular, the loss decreases from 1 to 0.13. Based on the train loss, many reconstruction loss values are calculated [73]. The best reconstruction loss helps to find the optimal threshold. Accordingly, we detect elephants by reconstructing the input traffic. The threshold value for elephant flow detection is 0.1555 with model validation accuracy of 97.95%, as shown in Table 5. If the reconstruction loss for a sample is greater than this threshold value, then we can infer that the model is seeing a pattern that it is not familiar with mouse flows. The test accuracy score obtained is 96.58% accuracy.

5.4.2. Discussion on Performance Result of Autoencoder. The loss function quantifies how well or how bad the given predictor is when it classifies the input data points in the dataset. The smaller the loss, the better the classifier is doing at modeling the relationship between the input data and

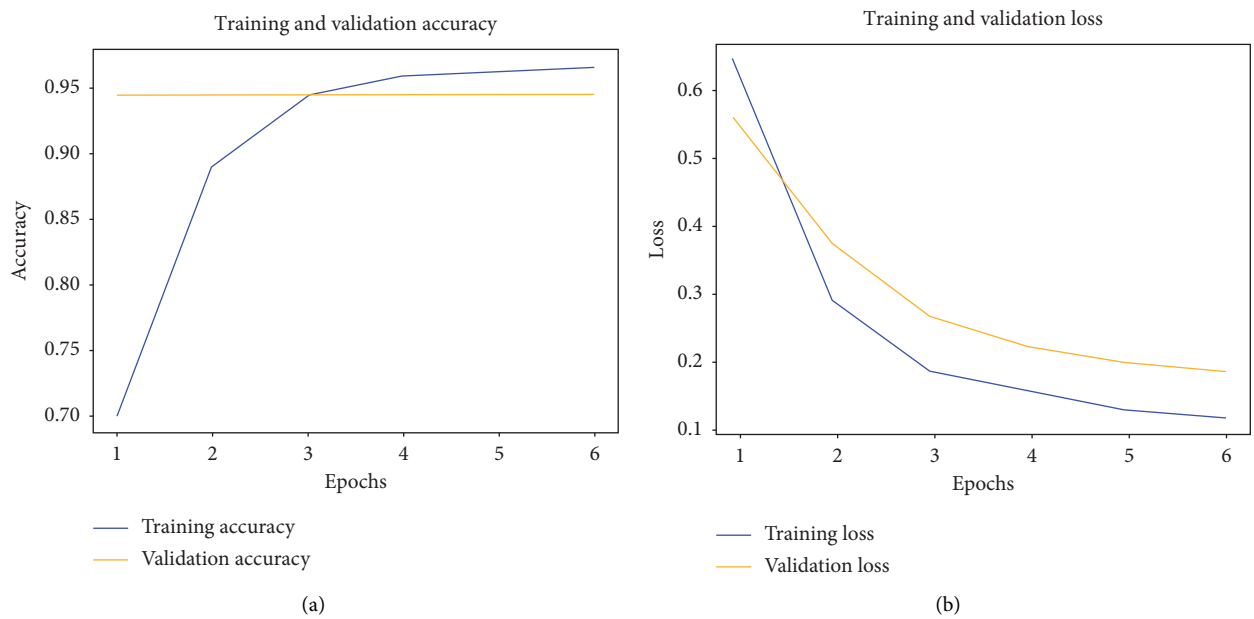

Figure 5: DNN loss on Unicauca-dataset [69]: (a) training accuracy and validation accuracy and (b) training loss and validation loss.

Figure 6: Confusion matrix on Unicauca-dataset.

output class labels [70]. Loss is a cumulative noise per epoch. At the beginning of each epoch, the loss is 9.5. For each time, loss calculation is added to the loss metric [70]. What is seen over time in the plotted results is the total loss of training and validation loss is decreasing generally. This decreasing of loss means the weights of the network is getting more accurate. Usually, reading such a low loss of near zero indicates that the potential of the model to detect elephant flows.

It is implicitly expected that the classification accuracy is inversely proportional to the average loss value, i.e., the training loss and validation loss both decrease and stabilize at a specific point at round epoch 20.

Usually, the validation loss is greater than the training loss. This may indicate that the model is under fitting for some extent [63]. Even though the results show small loss, result indicates that further training is needed to reduce the loss incurred during training for more performance improvement. Alternatively, we can also increase the training data either by obtaining more samples or augmenting the data [75].

(3) Experimenting Elephant Flow Detection Using CNN and LSTM Algorithms. We compared the performance of the DNN with the state of the art CNN and LSTM algorithms. We ran the CNN on the SDN dataset, and its performance result was 98.17% accuracy and 98.13% validation accuracy. Loss of 1.83% is occurred during training. The total training takes 8 seconds to build the elephant detection model. We also develop the elephant detection model using LSTM. The training performance result scores 98.78% training accuracy and 97.55% validation accuracy, respectively, as shown in Figure 10(a). The training takes

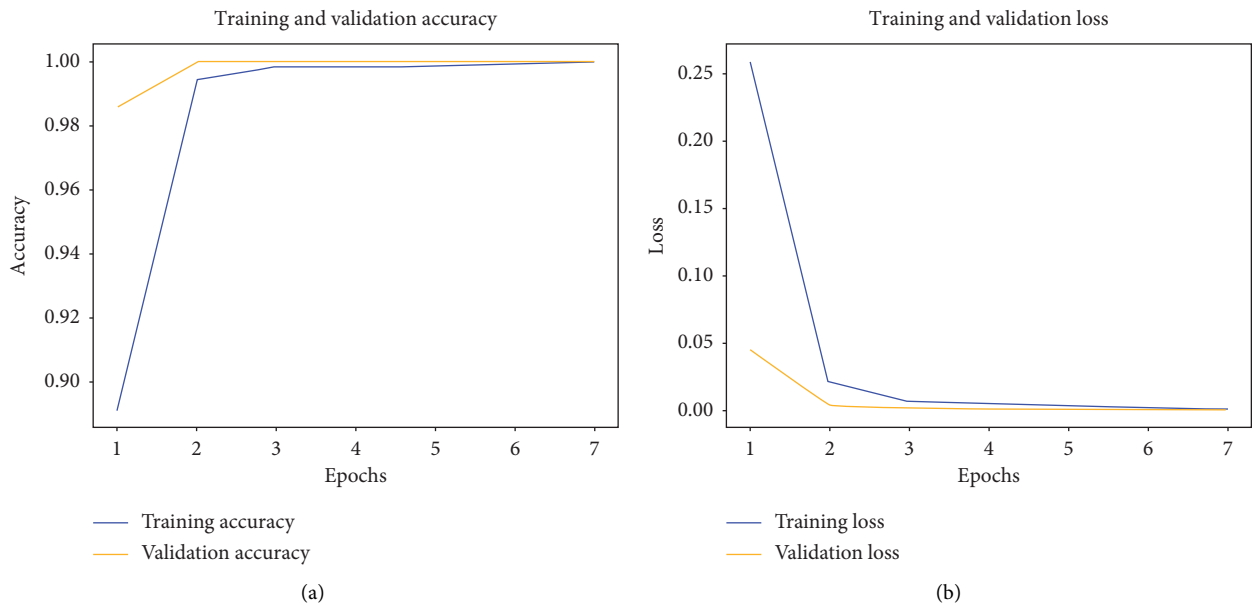

Figure 7: DNN loss on SDN dataset [12]: (a) training accuracy and validation accuracy and (b) training loss and validation loss.

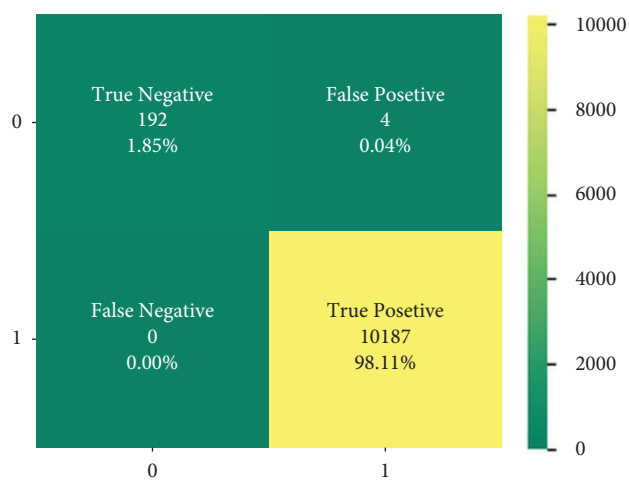

Figure 8: 2×2 confusion matrix SDN dataset using DNN.

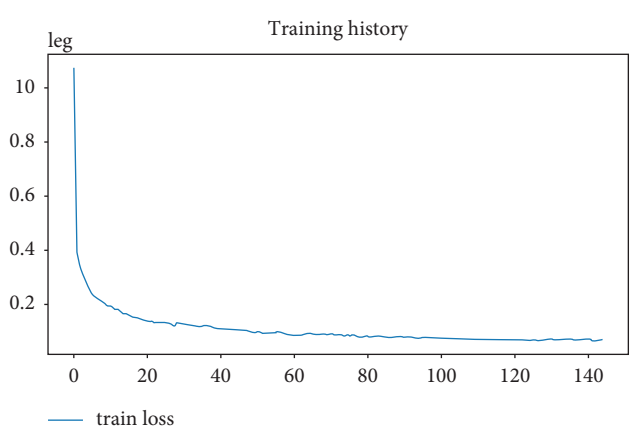

Figure 9: Loss visualization autoencoder.

Table 5: Best validation accuracy and threshold.

Best validation accuracy: 0.9795 for threshold: 0.1555
Perentile: 90, threshold: 0.1005, validation accuracy: 0.9521
Perentile: 91, threshold: 0.1028 validation, accuracy: 0.9543
Perentile: 92, threshold: 0.1083, validation accuracy: 0.9658
Perentile: 93, threshold: 0.1113, validation accuracy: 0.9658
Perentile: 94, threshold: 0.1173, validation accuracy: 0.9703
Perentile: 95, threshold: 0.1245 validation accuracy: 0.9726
Perentile: 96, threshold: 0.1383, validation accuracy: 0.9772
Perentile: 97, threshold: 0.1555, validation accuracy: 0.9795
Perentile: 98, threshold: 0.192, validation accuracy: 0.9795
Perentile: 99, threshold: 0.2614, validation accuracy: 0.8174
Best validation accuracy: 0.9795 for threshold: 0.1555.

57 seconds in average with minimum (3.2%) loss, as shown in Figure 10b.

5.4.3. Discussion on Performance Result of LSTM. A slightly higher increase of validation accuracy over training accuracy up to epoch 11 is due to the overfitting problem and stabilizes the training to the normal state after 11 epochs [63]. The LSTM model shows fairly increasing accuracy after 11 epochs. Thereby, the loss value drops from 7.84% to 0.56% which is near to zero. A low loss of near zero indicates that the elephant and mouse classifier has the model to categorize the traffic flow to the target traffic class [76].

At the beginning of the training, overfitting of the model on the SDN dataset took place. Figure 10(a) shows that after a certain epoch, the validation accuracy increases while the training accuracy decreases. After epoch 5, the training accuracy becomes higher than the validation accuracy, which is due to the dropout techniques we incorporated. Here, both the training accuracy and validation accuracy were balanced, which is advisable when building the best models. Figure 10(b) shows the gap between training loss

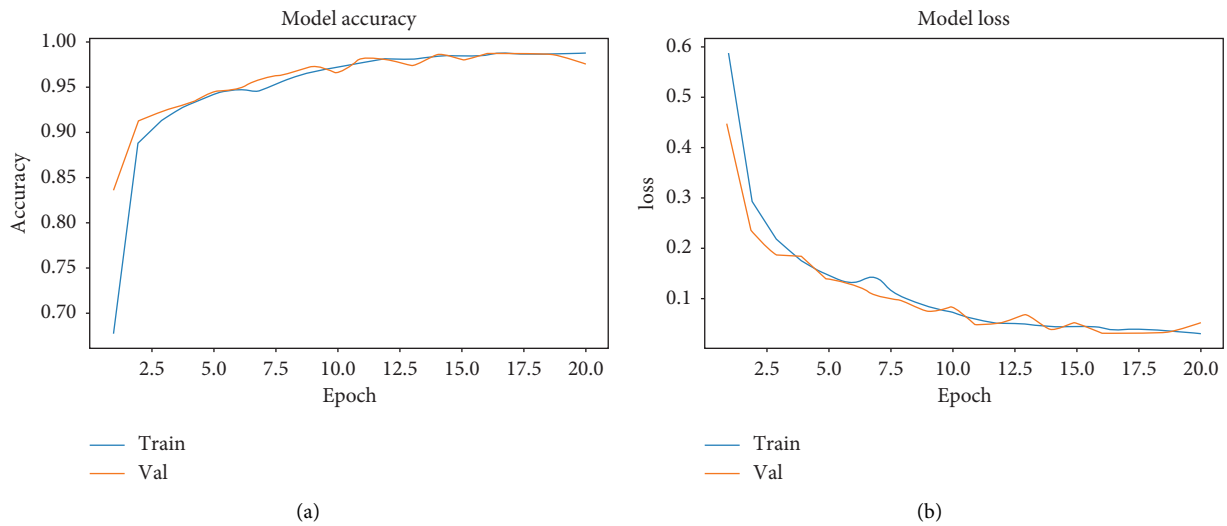

Figure 10: LSTM model performance: (a) training accuracy and validation accuracy and (b) training loss and validation loss.

and validation loss over time. Both the training loss and validation loss are close to zero. The scenario ensures that the training was performed in a normal situation that is free from overfitting and underfitting problems. Therefore, LSTM shows good fitting, validation accuracy, and training accuracy are high, with the former slightly lower than the latter [63]. LSTM not only shows good fit, but also achieves the highest accuracy (98.78%) among the other deep learning algorithms used in this work.

Experiment 2. Comparison of DL techniques

Deep learning algorithms were compared in terms of their relevant complexity including performance in accuracy, model runtime, number of trainable parameters, and loss incurred.

The optimal performance accuracy of the models is represented in various indicator parameters. 99.12%, average accuracy, is the highest performance obtained with the DNN. Although this is the highest performance, LSTM, CNN, and autoencoder also provide promising models with accuracy greater than 96%.

In addition to accuracy, we also compared the time taken to produce the models, as shown in Table 6. From the comparison results, our models generally run in less than 60 seconds. In particular, the DNN has the lowest runtime of all the algorithms used, at 4 seconds. Deep autoencoder takes 59 seconds, which is a relatively slow performance due to the additional compression and decompression tasks [20, 77].

The comparison result on three QoS-based datasets, NIMIS, Unicauca, and SDN, is also presented with the number of parameters being, 24, 75, and 21, respectively. Thus, the dataset preparation and parameterizations were important steps in the elephant flow detection process. Duration, flow size, packet size, and application type were used as heuristics parameters. Since the focus of this work is on SDN, the SDN dataset with the DNN, CNN, LSTM, and deep autoencoder was used.

Experiment 3. What optimal epochs will be required for producing the flow classification model?

As we try to find the optimal or correct number of epochs to train a neural network model, we experiment with different epoch numbers and batch sizes to check how accuracy is affected. It is also used to check whether overfitting and undermining of the training occurs when classifying the network traffic. First, we perform the training on the NIMS dataset with 2 hidden layers, an input layer and an output layer, by initializing the epochs to 50.

Epoch is a kind of hyperparameter that plays an essential role in the training process of a model and helps to decide whether the data are overtrained or not [67]. Thus, we find that epochs play an important role in obtaining good accuracy for training the neural network model only on the traffic training dataset [70].

Neural networks are able to learn by changing the distribution of weights. It is possible to approximate a function that is representative of the patterns in the input. The key idea is to stimulate the black box with new stimuli (data) until a sufficiently well-structured representation is obtained [78]. Therefore, the dataset is tested in different epochal iterations. Accordingly, we test different epoch intervals to find the optimal epoch for the dataset NIMS using CNN-based AE, as shown in Table 7.

We choose CNN-based autoencoder to assess the effect of epoch on the models having the behavior of mixed algorithms used to construct models in the work: CNN and AE. The result of this mixed algorithm is shown in Table 7. We can see the accuracy of the models for the six different epoch values. It is shown that the classification result is trained with 5, 10, 20, 50, 100, and 1000 epochs, respectively, with constant stack size. The network computes the errors for both the training and validation sets. We stop training when the validation error reaches the minimum.

Table 6: Comparison of deep learning algorithm.

Algorithm	Performance in accuracy	Model run time	Loss	Dataset used	#Parameters
DNN	99.12% in average	4 s	0.041	NIMS, Unicauca, and SDN	NIMS-24, SDN-21, and Unicauca-75
CNN	98.49%	5 s 3 ms	0.015	SDN	21
LSTM	98.78%	57 s 60 ms	0.03	SDN	21
Deep autoencoder	97.95%	59 s	0.13	SDN	21

Table 7: Model performance using different epochs.

Batch size	Epoch	Accuracy NIMS dataset [30] (%)
512	5	96.08
512	10	97.01
512	20	98.70
512	50	98.74
512	100	98.70
512	1000	98.06

6. Discussion

As we tried to find the optimal epochs to train a deep learning, an experiment was conducted with various numbers of epochs to check how the accuracy affects the model performance as well as if there is any overfitting happening in the training process.

As the results shown in Table 7, as the epoch value increases, the accuracy of the system is improved starting epoch 5 to epoch 50.

We can conclude that the accuracy of the model is promising for detecting elephant flows with the same stack size and number of classes; so most models have good accuracy for each epoch [76]. However, performance begins to decline after epoch 50. This is due to the fact that up to epoch 50, most records are classified into their category based on the optimal updated weights. Thus, the classification saturates when epoch is selected until about epoch 50, then the classification accuracy decreases because the traffic flows are already assigned to their class, i.e., the updated weights are summed up beyond the expected computational result [70]. In other words: If the accuracy of the training data increases, but the accuracy of the validation data remains the same or even decreases, it means that the model is overfitted, which in turn means that we should stop the training process [78]. Thus, we can conclude that the validation error is increasing. At epoch 1024, the accuracy becomes high (98.22%), but this is an overfitting since the training error is equal to 0 [70]. So we have to stop at epoch 50 for sure. We can stop the training process early at about epoch 50 to get better performance of the model. Since epoch 50 has achieved 98.74% accuracy, this is optimal and promising for integration with SDN controllers or OpenSwitches for traffic management to improve QoS.

Experiment 4. By what extent batch size variation affects traffic flow classification model at constant epoch?

Table 8: Different batch size on model performance.

Batch size	Epoch	Accuracy
32	50	97.01
64	50	94.30
128	50	97.60
512	50	98.74
1024	50	98.22

We test the performance of the elephant detection models when the lot size varies. As shown in Table 8, the best model performance is achieved at a batch size of 512, while the epoch is constant at 50 when using the CNN-based AE.

The batch size indicates how much of the dataset is used for each training step. The training process is a stepwise process where the dataset is divided into batches [76]. In our case, we tested our dataset by dividing it into 32, 64, 512, and 1024 batches. The stack size of 512 achieves the best performance, 98.74% accuracy, as seen in Table 8 mentioned above. We tested the variation of stack size to understand its impact on the traffic model construction. We tested the stack size starting from 32, 64, 128, 512, and 1024 under the constant optimal number of 50 epochs. The stack size with 50 epochs gives an accuracy of 98.74%. The best results in traffic classification accuracy are obtained with stack size of 512 and 1024 examples. The larger the stack size, the higher the traffic classification accuracy [79]. It can be concluded that a model with a stack size of 512 examples and an epoch of 50 examples is promising for a small traffic dataset and shows the potential of the QoS provisioning mechanism.

7. Conclusion and Future Work

7.1. Conclusion. Deep learning techniques have become one of the most interesting and practical topics in network engineering. In this paper, we propose a traffic classification model that detects elephant flows and can be integrated with SDN controllers to ensure good QoS. In particular, we used deep neural networks, CNN, LSTM, and autoencoders. To consider a model as the best model, its performance was tested with the training dataset, validation dataset, and test datasets. The flow classification model was found to be the most influential model for classifying elephant and mouse flows in the SDN. The average detection rate of elephant fluxes is 98.77%, 98.17%, and 98.78% using DNN, CNN, and LSTM in the three datasets, respectively. Therefore, we can conclude that the potential and capabilities of deep learning algorithms for elephant flow detection are promising for better QoS in the SDN.

Therefore, we can conclude that the potentiality and promising of deep learning algorithms on elephant flow detection for better QoS in the SDN.

7.2. Research Implication. Deep learning techniques have become one of the most interesting and practical topics in network engineering. In this paper, we propose a traffic classification model that detects elephant flows and can be integrated with SDN controllers to ensure good QoS. In particular, we used deep neural networks, CNN, LSTM, and autoencoders. To consider a model as the best model, its performance was tested with the training dataset, validation dataset, and test datasets. The flow classification model was found to be the most influential model for classifying elephant and mouse flows in the SDN. The average detection rate of elephant fluxes is 98.77%, 98.17%, and 98.78% using DNN, CNN, and LSTM in the three datasets, respectively. Therefore, we can conclude that the potential and capabilities of deep learning algorithms for elephant flow detection are promising for better QoS in the SDN.

7.3. Future Work. In our future work, we plan to conduct a research on the SDN by integrating the best model obtained in this work so that we will review the QoS and QoE improvements between users and network administrators. In the future, we plan to increase the amount of data and extend this research study by using different deep learning methods in an SDN environment. This will provide an opportunity for very accurate, fast, and reliable classification. In particular, the elephant flow detection task should be better tested with a generative adversarial network (GAN) since GAN shows good performance in pattern recognition. Generating adversarial deep convolutional networks helps for effective fits and expands traffic dataset to maintain balance between elephant and mice classes of the dataset, which enhances the dataset stability [36]. In addition, the effect of explainable artificial intelligence (EAI) to improve the quality of service in the SDN networks is investigated [11, 12, 80–82].

Data Availability

The datasets used to support the findings of this study have been deposited as NIMS QoS dataset : https://projects.cs.dal.ca/projectx/Download.htmlUnicauca_Dataset:https://www.kaggle.com/datasets/jsrojas/ip-network-traffic-flows-labeled-with-87-apps and SDN dataset: https://github.com/getahunwassie/Researches-papers.git.

Disclosure

Prof. Jianguo Ding is the author's PhD principal supervisor.

Conflicts of Interest

The authors declare that they have no conflicts of interest. The author is a PhD student in Addis Ababa University.

Acknowledgments

The authors thank the authors of the dataset [52, 69, 83] who prepared traffic datasets and made it available openly.

References

[1] J. Falk, H. Geppert, F. Durr, S. Bhowmik, and K. Rothermel, "Dynamic QoS-aware traffic planning for time-triggered flows in the real-time data plane," *IEEE Transactions on Network and Service Management*, vol. 19, no. 2, pp. 1807–1825, 2022.

[2] A. T. Oliveira, "SDN-based architecture for providing QoS to high performance distributed applications," in *Proceedings of the 2018 IEEE Symposium on Computers and Communications (ISCC)*, Natal, Brazil, November 2018.

[3] H. A. H. Ibrahim, S. Mohd Nor, and A. Ahmed, "Internet traffic classification algorithm based on hybrid classifiers to identify online games traffic," *Jurnal Teknologi*, vol. 64, no. 3, 2013.

[4] B. Khalida and S. Chaveesuk, "Bilal KhaContinuance intention to use digital payments in mitigating the spread of COVID-19 virus," *International Journal of Data and Network Science*, vol. 6, 2022.

[5] P. W. Singha, "Cloud computing classroom acceptance model in Thailand higher education's institutes A conceptual framework," in *Proceedings of the 2018 10th International Conference on Information Mana*, Salford UK, September 2018.

[6] A. Gholamhosseinian, "QOS for multimedia applications with emphasize on video conferencing," *Modern Communication System and Networks*, Halmstad University, School of IDE, Halmstad, Sweden, 2011.

[7] C. Wang and H. Kim, "Touchdown on the cloud:the impact of the super bowl on cloud," in *Proceedings of the 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService)*, Newark, CA, USA, September 2019.

[8] M. Feng, "Hierarchical video frame sequence representation with deep convolutional graph network," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, Munich, Germany, September 2018.

[9] Y. Zhai, H. Xu, H. Wang, Z. Meng, and H. Huang, "Joint routing and sketch configuration in software-defined networking," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2092–2105, 2020.

[10] A. Chhabra and M. Kiran, "Classifying elephant and mice flows in high-speed scientific networks," in *Proceedings of the Division of Electronics and Communication Engineering*, Denver, USA, November 2017.

[11] M. Hamdan, B. Mohammed, U. Humayun et al., "Flow-aware elephant flow detection for software-defined networks," *IEEE Access*, vol. 8, pp. 72585–72597, 2020.

[12] Fan and R. Liu, "Investigation of machine learning based network traffic classification," in *Proceedings of the International Symposium on Wireless Communication Systems (ISWCS)*, Bologna, Italy, August 2017.

[13] D. Shi, "Online encrypted Skype identification based on an updating mechanism," 2022, https://arxiv.org/abs/2203.12141.

[14] M. J. Siavoshani, R. S. H. Zade, M. Saberian, and M. Lotfollahi, "Deep packet: a novel approach for encrypted trafic classification using deep learning," 2018, https://arxiv.org/abs/1709.02656.

[15] S. Adibi, "Traffic classification , packet-flow, and application-based approaches," *International Journal of Advanced Computer Science and Applications*, vol. 1, 2010.

[16] S. Dong, X. Zhang, and D. Zhou, "Auto adaptive identification algorithm based on network traffic flow," *International*

Journal of Computers Communications & Control 1841-9836, vol. 9, no. 6, pp. 672–685, 2014.

[17] B. Akbari, "Optimal QoS-aware network reconfiguration in software defined cloud data centers," *Computer Networks*, vol. 120, 2017.

[18] Y. Harchol, D. Hay, Y. Koral, and A. Bremler-Barr, "Deep packet inspection as a service," in *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, Sydney Australia, December 2014.

[19] D. Z. Shi and W. D. Dong, "Traffic classification model based on integration of multiple classifiers," *Journal of Computational Information Systems*, vol. 8, 2012.

[20] S. Dong, "Multi class SVM algorithm with active learning for network traffic classification," *Expert Systems with Applications*, vol. 176, Article ID 114885, 2021.

[21] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Computer Science Review*, vol. 40, Article ID 100379, 2021.

[22] H. Zhu and X. Jin, "Neural packet classification," 2019, https://arxiv.org/abs/1902.10319.

[23] D. Aloraifan, I. Ahmad, and E. Alrashed, "Deep learning based network traffic matrix prediction," *International Journal of Intelligent Networks*, vol. 2, pp. 46–56, 2021.

[24] M. A. Jabbar and S. R. Tiwari, "Foundations of deep learning and its applications to health informatics," in *Deep Learning in Biomedical and Health Informatics,* CRC Press, Boca Raton, FL, USA, 2021.

[25] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, "A survey of the recent architectures of deep convolutional neural networks," *Artificial Intelligence Review*, vol. 53, no. 8, pp. 5455–5516, 2020.

[26] C. rançois, *Deep Learning with Python*, Manning pub, Shelter Island, NY, USA, 2018.

[27] Y. Wei, "LSTM-autoencoder based anomaly detection for indoor air quality time series data," 2022, https://arxiv.org/abs/2204.06701.

[28] T. Tayeh, *An Attention-Based ConvLSTM Autoencoder with Dynamic Thresholding for Unsupervised Anomaly Detection in Multivariate Time Series*, ECE Department, Western University, London, UK, 2022.

[29] G. Bovenzi, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "A big data-enabled hierarchical framework for traffic classification," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 4, pp. 2608–2619, 2020.

[30] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescape, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, 2019.

[31] B. Pablo, "SDN-based overlay networks for QoS-aware routing," in *Proceedings of the 2016 workshop on Fostering Latin-American Research in Data Communication Networks*, Florianopolis Brazil, August 2016.

[32] H. Rutvij, R. Tan, and S. V. Ramani Jhaveri, "Real-time QoS-aware routing scheme in SDN-based robotic cyber-physical systems," in *Proceedings of the 2019 IEEE 5th International Conference on Mechatronics System and Robots (ICMSR)*, IEEE, Singapore, September 2019.

[33] Noora and R. E. Overill Al Khater, "Network traffic classification techniques and challenges," in *Proceedings of the 2015 Tenth international conference on digital information management (ICDIM)*, IEEE, Jeju, Korea (South), January 2015.

[34] O. M. A. Alssaheli, Z. Z. Abidin, N. A. Zakaria, and Z. A. Abas, "Software defined network based load balancing for network performance evaluation," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 4, 2022.

[35] A. Nascita, A. Montieri, G. Aceto, D. Ciuonzo, V. Persico, and A. Pescape, "XAI meets mobile traffic classification: understanding and improving multimodal deep learning architectures," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4225–4246, 2021.

[36] S. Dong, Y. Xia, and T. Peng, "Traffic identification model based on generative adversarial deep convolutional network," *Annals of Telecommunications*, vol. 77, no. 9-10, pp. 573–587, 2021.

[37] M. Ali, "Intelligent SDN traffic classification using deep learning," in *Proceedings of the Campus Conference Paper*, Nagoya, Japan, May 2020.

[38] S. Dong, Y. Xia, and T. Peng, "Network abnormal traffic detection model based on semi-supervised deep reinforcement learning," *IEEE Transactions on Network and Service Management*, vol. 18, no. 4, pp. 4197–4212, 2021.

[39] S. Dong and Y. Xia, "Network traffic identification in packet sampling environment," *Digital Communications and Networks*, 2022.

[40] L. Chang, "Fs-net: a flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE INFOCOM 2019-IEEE Conference On Computer Communications*, Paris, France, June 2019.

[41] M. Basit, "An efficient internet traffic classification system using deep learning for IoT," 2021, https://arxiv.org/abs/2107.12193.

[42] S. Dong and R. Li, "Traffic identification method based on multiple probabilistic neural network model," *Neural Computing and Applications*, vol. 31, no. 2, pp. 473–487, 2019.

[43] W. J. Niloofar and D. Liu Bayat, "Deep learning for network traffic classification," 2021, https://arxiv.org/abs/2107.12193.

[44] R. T. K. Tamil Selvi, "Deep learning based traffic classification in software defined networking," *International Journal Of Scientific and Technology Research*, vol. 9, no. 2, 2020.

[45] H. Ren-Hung, "An LSTM-Based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level," *Applied Sciences*, vol. 9, p. 3414, 2019.

[46] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret, "Conditional variational autoencoder for prediction and feature recovery applied to intrusion detection in iot," *Sensors*, vol. 17, no. 9, p. 1967, 2017.

[47] E. F. Franco, P. Rana, A. Cruz et al., "Performance comparison of deep learning autoencoders for cancer subtype detection using multi-omics data," *Cancers*, vol. 13, no. 9, p. 2013, 2021.

[48] E. Pintelas, I. E. Livieris, and P. E. Pintelas, "A convolutional autoencoder topology for classification in high-dimensional noisy image datasets," *Sensors*, vol. 21, p. 7731, 2021.

[49] L. T. Ibrahim, "Online traffic measurement and analysis in big data: comparative research review," *American Journal of Applied Sciences*, vol. 13, no. 4, pp. 420–431, 2016.

[50] D. Shi, W. Liu, and Y. Cui, "Feature selection algorithm based on correlation between muti metric network traffic flow features," *The International Arab Journal of Information Technology*, vol. 14, 2017.

[51] Q. Zhang, "A classification supervised auto-encoder based on predefined evenly-distributed class centroids," *School of Communication and Information Engineering*, Shanghai University, Shanghai, China, 2019.

[52] R. Alshammari and A. N. Zincir-Heywood, "Can encrypted traffic be identified without port numbers, IP addresses and payload inspection?" *Computer Networks*, vol. 55, no. 6, pp. 1326–1350, 2011.

[53] zgzhengSEU, "Nicauca Version2 dataset," 2021, https://www.kaggle.com/code/kerneler/starter-ip-network-traffic-flows-49778383-1.

[54] B. Marcus Vinicius, "Identifying elephant flows using dynamic thresholds in programmable IXP networks," *Journal of Internet Services and Applications*, vol. 11, 2020.

[55] G. Tam, *Cisco Application Centric Infrastructure*, Americas Headquarters, Cisco Systems, Inc, San Jose, CA, USA, 2018.

[56] P. Samira, *A Survey on Deep Learning: Algorithms, Techniques, and Applications*, ACM Computing Surveys (CSUR), New York, NY, USA, 2018.

[57] P. Amudha and S. Sivakumari Amitha Mathew, "Deep learning techniques: an overview," in *Proceedings of the International conference on advanced machine learning technologies and applications*, Cairo, Egypt, July 2020.

[58] Ons and K. Piamrat, "Decision tree-based blending method using deep-learning for network management," in *Proceedings of the NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, Budapest, Hungary, January 2022.

[59] C. Xinyi and F. W. Hu, "CLD-Net: a network combining CNN and LSTM for internet encrypted traffic classification," *Security and Communication Networks*, vol. 2021, Article ID 5518460, 15 pages, 2021.

[60] K. Piamrat, "Intelligent traffic management in next-generation networks," *Future Internet*, vol. 14, 2022.

[61] D. Piga, "A neural network architecture for learning dynamical systems," *IDSIA Dalle Molle Institute for Artificial Intelligence*, vol. 35, 2021.

[62] S. Malpe, *Automated Leaf Disease Detection and Treatment Recommendation Using Transfer Learning*, National College of Ireland, Dublin, Ireland, 2019.

[63] J. Bilmes, *Underfitting and Overfitting in Machine Learning*, Javatpoint, Noida, India, 2020.

[64] N. Koenigstein, "Autoencoder," 2020, https://arxiv.org/abs/2003.05991.

[65] H. Xu and B. Li, *TinyFlow: Breaking Elephants Down into Mice in Data Center Networks*, IEEE, Toronto, Canada, 2014.

[66] E. F. Franco, P. Rana, A. Cruz et al., "Performance,Comparison of deep learning autoencoders for cancer subtype detection using multi-omics data," *Cancers*, vol. 13, no. 9, 2021.

[67] T. Fischer and C. Krauss, "Deep learning with long short-term Memory networks for financial market predictions," *European Journal of Operational Research*, vol. 270, no. 2, pp. 654–669, 2018.

[68] H. V. Pham, S. Qian, J. Wang et al., "Problems and opportunities in training deep learning software systems: an analysis of variance," in *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*, Australia, January 2020.

[69] K. Piamrat, "Network traffic classification using machine learning for software defined networks," in *Proceedings of the IFIP International Conference on Machine Learning for Networking (MLN'2019)*, Paris, France, December 2019.

[70] S. Rao Saahil Afaq, "Significance of epochs on training A neural network," *International Journal Of Scientific and Technology Research*, vol. 9, no. 6, 2020.

[71] M. O. Khan, V. Toro Arana, C. Rubbert et al., "Deep learning for safe autonomous driving: current challenges and future directions," *Journal of Neurosurgery*, vol. 99, pp. 1–11, 2020.

[72] B. Ru, "Speedy performance estimation for neural architecture search," 2021, https://arxiv.org/abs/2006.04492.

[73] M. W. Theunissen, H. D. Marelie, and A. E. W. Venter, "Pre-interpolation loss behaviour in neural networks," 2021, https://arxiv.org/abs/2103.07986.

[74] Y. H. Park, "Concise logarithmic loss function for robust training of anomaly detection model," 2022, https://arxiv.org/abs/2201.05748.

[75] L. Suhua, "How training data affect the accuracy and robustness of neural networks for image classification," 2018, https://openreview.net/forum?id=HklKWhC5F7.

[76] A. Taner, Y. B. Öztekin, and H. Duran, "Performance analysis of deep learning CNN models for variety classification in hazelnut," *Sustainability*, vol. 13, no. 12, p. 6527, 2021.

[77] Quentin and D. A. Fournier, "Empirical comparison between autoencoders and traditional dimensionality reduction methods," in *Proceedings of the 2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, IEE, Sardinia, Italy, June 2019.

[78] A. Sengupta, "A review of deep learning with special emphasis on architectures, applications and recent trends," *IEEE Transaction on XXX*, vol. 20, 2019.

[79] P. M. Radiuk, "Impact of training set batch size on the performance of convolutional neural networks for diverse datasets," *Information Technology and Management Science*, vol. 20, no. 1, 2017.

[80] Y. Chia, "Predicting drug response of tumors from integrated genomic profiles by deep neural network," *BMC Medical Genomics*, vol. 12, no. 1, pp. 143–155, 2019.

[81] L. D. Peter, "An introduction to computer networks," 2020, https://intronetworks.cs.luc.edu/.

[82] L. Zhu and P. Spachos, *Towards Image Classification with Machine Learning Methodologies for Smartphones*, University of Guelph, Guelph, Canada, 2019.