International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING

ISSN:2147-6799 **www.ijisae.org** **Original Research Paper**

A Deep Learning Based System for Traffic Engineering in Software Defined Networks

Sudad Abdulrazzaq*[1], Mehmet Demirci[2]

Submitted: 23/09/2020 **Accepted : 09/12/2020**

Abstract: Traffic engineering is essential for network management, particularly in today's large networks carrying massive amounts of data. Traffic engineering aims to increase the network's efficiency and reliability through intelligent allocation of resources. In this paper, we propose a deep learning-based traffic engineering system in Software-Defined Networks to improve bandwidth allocation among various applications. The proposed system conducts traffic classification based on deep neural network and one dimensional convolution neural network models. It aims to improve the Quality of Service by identifying flows from various applications and distributing the identified flow to multiple queues where each queue has a different priority. Then, it applies traffic shaping in order to manage network bandwidth and the volume of incoming traffic. To increase the network's performance and avoid traffic congestion, we implement a technique that considers the port capacity to accomplish general load balancing. We solved the issue of imbalanced dataset by implementing an oversampling technique called Synthetic Minority Over-Sampling Technique. The performance of DNN and 1-D CNN have been compared and evaluated with some of machine learning models, such as KNN, SVM, DT, and RF. The results showed 1-D CNN and DNN are able to achieve more than %88 accuracy of traffic captured in 5s and 10s timeout, while KNN and RF are able to achieve more than %98 accuracy of traffic captured in 15s and 30s timeout, and the evaluation of the overall system showed applying traffic shaping to the identified flow increases the network's performance and bandwidth availability.

Keywords: *Deep Learning, Quality of Service, Software Defined Networks, Traffic Classification, Traffic Engineering, Traffic Shaping*

1. Introduction

In recent decades, enterprise networks have been experiencing a massive increase in the amount of data handled and made available to customers, employees, and business partners. As a result, resource usage and network flow planning became an important network management practice. Traffic engineering (TE) is a crucial element of network management in today's network. Traffic engineering aims to increase the network's efficiency and reliability through the intelligent allocation of resources.

Software-Defined Networking (SDN) is a network architecture that enables the separation of the data plane from the control plane. The forwarding devices in the data plane are managed by the control plane. One of the essential aspects of SDN is that network intelligence is logically centralized in one place, which maintains and provides a global view of the network. The SDN controller acts as the network's brain and talks with the forwarding devices through OpenFlow protocol. SDN aims to simplify network management, improve resource utilization, promote innovation, and optimize networks' performance [1].

Traffic classification is a technique that deals with the problems of designating network traffic to a previously defined group of classes based on their properties or their behaviors. Traffic classification is crucial for network management, and it is widely used in different networking fields such as Quality of Service (QoS), flow planning, network security, resource provisioning, performance monitoring. There are three commonly used solutions in traffic classification: end-user application, traffic classes, and application protocols. After network traffic is classified, any combination of QoS tools can be applied based on the enterprise network's requirements [2].

The most common and widely used traffic classification techniques are port-based, payload-based, and statistics-based.

The port-based technique is considered one of the simplest techniques because it depends on mapping applications to port numbers. However, this technique may be ineffective since many applications are increasingly using random or dynamic port numbers. A payload-based technique, also known as Deep Packet Inspection (DPI), has been presented to overcome the port-based technique's problems. DPI works by checking packet payloads and comparing them with the signatures of known protocols. This technique can perform classification accurately, but there are some downsides: encrypted traffic can be challenging to deal with, it requires intensive computation and manual signature maintenance. Another technique is statistics-based, which relies on time series or statistics of traffic flows. It is capable of handling both encrypted and unencrypted traffic. This approach generally utilizes machine learning (ML) algorithms, whose performance is highly dependent on engineered features, limiting their generalizability. The statistics-based approach also utilizes deep learning (DL) algorithms, which overcome ML's limitations. Deep Learning algorithms can automatically select features through training and learn very complicated patterns that make DL useful for traffic

[1] *Dept. of Computer Sci., Informatics Institute, Gazi University Ankara – 06560, Turkey*
ORCID ID: 0000-0002-1575-0248
[2] *Dept. of Computer Eng., Gazi University, Ankara – 06570, Turkey*
ORCID ID: 0000-0002-1088-5215
* *Corresponding Author Email: sudad.alajili@protonmail.com*

classification [3] [4].

In this paper, we propose a deep learning based traffic classification system based on deep neural network (DNN) and one dimensional convolution neural network (1-D CNN) for traffic engineering in SDN. Time-based statistical features of the flow are used to train the models. The classified flow is tagged with the ToS field and sent to the controller through the OpenFlow protocol. The proposed system works to improve the Quality of Service of the identified flows by distributing the identified flow to multiple queues where each queue has a different priority value. Then, it applies the traffic shaping technique, which manages network bandwidth by controlling the rate and the volume of incoming flows to avoid congestion. A load balancing technique is installed on the OpenFlow switch that increases the network's performance. It tries to accomplish a general load balancing by considering the capacity of each egress port.

The rest of the paper is organized as follows. Section 2 presents an overview of the related works. The proposed system is explained in Section 3. The evaluation of deep learning models and the overall system is shown in Section 4. Finally, we conclude the paper in Section 5.

2. Related Work

In this section, we present some works to discuss traffic engineering and traffic classification in SDN. Traffic engineering is an import mechanism and has been one of the hot topics to improve Internet performance for decades. Karakus et al. [5] presented a detailed QoS study in OpenFlow enabled networks by surveying the QoS motivated studies. It gave an overview of the relations between QoS and SDN. The studies considered are related to the following topics: inter-domain routing, multimedia flows routing, queue management, and scheduling, resource reservation, Quality of Experience (QoE) awareness, network monitoring, and other QoS-centric mechanisms. The study concludes that QoS can benefit from the concept of SDN. It also discusses the QoS capabilities of OpenFlow and outlines the potential challenges and problems that need to be addressed for better QoS capabilities.

Jeong et al. [6] proposed a system that combines traffic engineering and DPI in SDN. The proposed system aims to enhance the Quality of Service by using Deep Packet Inspection based traffic classification to identify traffic flows, and the identified flows are being allocated to multiple queues. The system enables the network administrator to set a mapping table between the identified flow and each queue. The results showed an increase in throughput and reduction in packet delay for identified flows. Yan et al. [7] presented a detailed study on the techniques used for traffic classification and reviewed commonly used machine learning algorithms in traffic classification in SDNs. They also presented some general challenges from a traffic classification perspective and recommendations on traffic classification in SDN. Amaral et al. [8] proposed a simple architecture for collecting traffic based on SDN, and it was deployed in an enterprise environment. Internet traffic was collected using OpenFlow, and ensemble learning algorithms were applied to the gathered traffic. The results showed that these algorithms could obtain high accuracy, and the proposed system is suitable for small enterprise networks to identify traffic flows of interest.

Lashkari et al. [9] proposed an approach to detect and characterize Tor traffic based on time analysis of the flows, and only time-based statistical features were chosen for the study. The authors showed that time-based features could be used to detect Tor traffic effectively, and these features can also be used to classify the traffic and identify different types of applications. Also, it showed that the flow timeout influences the classification efficiency.

Rezaei et al. [10] presented a general framework for deep learning based traffic classification and provided general guidelines for the traffic classification process, such as data collection, feature selection, and model selection. The authors demonstrated deep learning applications for traffic classification and discussed some open problems and traffic classification challenges. Choubey et al. [11] proposed a traffic classification system based on artificial neural network and one dimensional convolutional neural network models. The proposed method used a dataset generated by Andrew Moore, where the data was collected from two different sites. The proposed system results showed that both models gave high accuracy without being exposed to overfitting, and the testing times of these models were less compared to machine learning models. Xu et al. [12] proposed a network architecture that combines a traffic classification system based on deep learning and SDNs. The traffic classification system was deployed as a virtualized network function (VNF) to reduce the load on the SDN controller and ensure that the failure of the VNF will not have a paralyzing effect on the network. The proposed system aims to improve QoS by assigning different network resources to different applications. The experiments showed that the proposed model performs better than existing classification algorithms, and the controller can assign proper route paths for different types of flows.

3. System Design and Implementation

Our system consists of a deep learning classifier which sits at the edge of the network that provides the controller with the classification result of the incoming traffic, a packet scheduler which controls the routing decisions, and a traffic monitoring module which regularly collects delay and bandwidth statistics from the physical network. Other modules can obtain the collected data.

3.1. Deep Learning Classifier

The deep learning classifier is a stand-alone server with GPU power to help in the training phase since this phase requires an unlimited number of inputs and outputs, and a large number of computing resources. The classifier is connected to Open vSwitch, which is located at the network's edge to capture and analyze the incoming packets.

The traffic classification process followed the deep learning based traffic classification framework introduced by [10]. The framework consists of six steps: data collection, pre-processing, feature selection, model selection, training and validation, and periodic model update.

3.1.1. Data Collection

There are numerous network traffic datasets available free online, but many of them are outdated. For our experiment, we will be using the Tor traffic dataset provided by Lashkari et al. [9]. This dataset has time-related statistical features.

There are eight types of traffic gathered from more than 18 representative applications and contains 23 time-related features.

The traffic types are audio-streaming, browsing, file transfer, instant messaging, e-mail, P2P, video-streaming, and VOIP. Moreover, the dataset contains traffic captured at different flow timeout 5, 10, 15, 30, 60, and 120 seconds. Table 1. lists and describes the features in the dataset.

Table 1. Features Description

Feature	Description
Forward Inter Arrival Time (Mean, Min, Max, Std)	The period among two packets sent in a forward way.
Backward Inter Arrival Time (Mean, Min, Max, Std)	The period among two packets sent in a backward way.
Flow Inter Arrival Time (Mean, Min, Max, Std)	The period among two packets sent in either way.
Active (Mean, Min, Max, Std)	Active time flow
Idle (Mean, Min, Max, Std)	The idle time of a flow
Fb Psec	Flow bytes sent per second
Fp Psec	Flow packets sent per second
Duration	Flow duration

3.1.2. Pre-processing

Preprocessing is a process that involves transforming raw data into a well-formatted dataset. It can affect classification performance significantly. The missing values were dropped from the dataset during the data cleaning stage. All the features are numerical; thus, there is no need to convert their values other than rescaling them. The target feature is converted to its numerical representation.
To make the classification process simpler, we decided to group the traffic of similar applications into different classes. Thus, we grouped the eight classes mentioned before into three classes based on priority where latency and bandwidth are considered important criteria. Table 2. lists the grouped classes and their priority.

Table 2. Traffic Classes

Class	*Application*	*Priority*
0	Chat, VOIP	High
1	Audio-Streaming, Browsing, E-mail	Moderate
2	File Transfer, P2P, Video-Streaming	Low

During grouping, the classes of 15 and 30 second flow duration datasets into three groups, the number of examples in Class 1 was relatively low compared to the other two classes. We have used Synthetic Minority Over-Sampling Technique (SMOTE) to resolve the imbalanced dataset issue.
Furthermore, the data has been scaled to improve the model's performance before starting the training process.
Normalization is the process where the values of all features are rescaled to be between zero and one. This the most commonly used scaling method for neural networks.

$$x_{norm} = \frac{x_i - min(x)}{max(x) - \min(x)} \quad (1)$$

Standardization is the process of rescaling all values of all features so that their mean is 0, and their standard deviation is equal to 1.

$$x_{standardized} = \frac{x_i - mean(x)}{std(x)} \quad (2)$$

3.1.3. Synthetic Minority Over-Sampling Technique

Synthetic Minority Over-Sampling Technique (SMOTE) is an over-sampling technique used to create synthetic examples for the minority class rather than replacing them. It was first inspired by a technique used in handwritten character recognition. The technique works on generating synthetic samples in a less application-specific way by operating in feature space.
The minority class is over-sampled by taking each minority class sample, then introducing the synthetic examples along the line segments joining any or all of the k minority class nearest neighbors. The k nearest neighbors are chosen at random based on the number of required samples.
Synthetic samples are generated by taking the difference between the considered feature vector and its nearest neighbor, multiply the difference by a random number 0 or 1, then add it to the considered feature vector. That leads to selecting a random point along the line segment between two specific features. The synthetic samples make the classifier create big and less specific decision regions rather than small and more specific regions [15].

3.1.4. Feature Selection

Feature selection is a process of identifying irrelevant or redundant features and remove them. Unneeded features may affect the performance of the model. The most commonly used feature for traffic classification is time series, header data, payload data, and statistical features.
All 23 features are considered to be relevant to the purpose of the study. As stated before, these features are statistical. The advantages of using statistical features are capable of handling both encrypted and unencrypted traffic since it relies on time series or statistics of traffic flows, and the computational complexity is low.

3.1.5. Model Selection

Many aspects will affect the choice of deep learning models for traffic classification. The important one is input features, they will affect the accuracy of the model, but it will also affect the input dimension, which affects computational complexity and the number of packets for classification. Also, the size of the dataset will affect the selection of the model. In the our study, deep neural networks and one dimensional convolution neural network were chosen to train the classifier, and were build from groud up.
Deep neural networks (DNNs), also known as multilayer perceptrons, are a collection of multiple perceptrons. Deep neural networks architecture consists of an input layer, one or more hidddn layers, and an output layer. Networks with over four layers are considered deep neural networks and are commonly used to solve complex and abstract data problems. The architecture of DNN that used to train the classifier consists of one input layer, three hidden layers, one output layer.
One-dimensional convolution neural network search to create models that use different groups of neurons to recognize various aspects of the data. These groups need to communicate with each other so that they can form the big picture. Its architecture consists of collection of multiple perceptrons, and one or more convolution layer. These layers are either interconnected or pooled. Convolutional Layer works on extracting features by using the matrix, and the result is a feature map. Pooling Layer reduces the features number while maintaining the important information. Spatial Pooling Layer, also known as downsamplin, will decreases the number of features while keeping the important information. After the convolution layers, fully connected neural network in used for the final task [20].
The architecture of 1D-CNN that used to train the classifier consists of one input layer, two of one-dimensional convolution layers, two of one-dimensional max pooling layers, one flatten layer, two of fully connected network layers, and one output layer.

Machine learning models, such as K-Nearest Neighbors, Support Vector Machine, Decision Tree, and Random Forest were used for comparison with deep learning models performance.

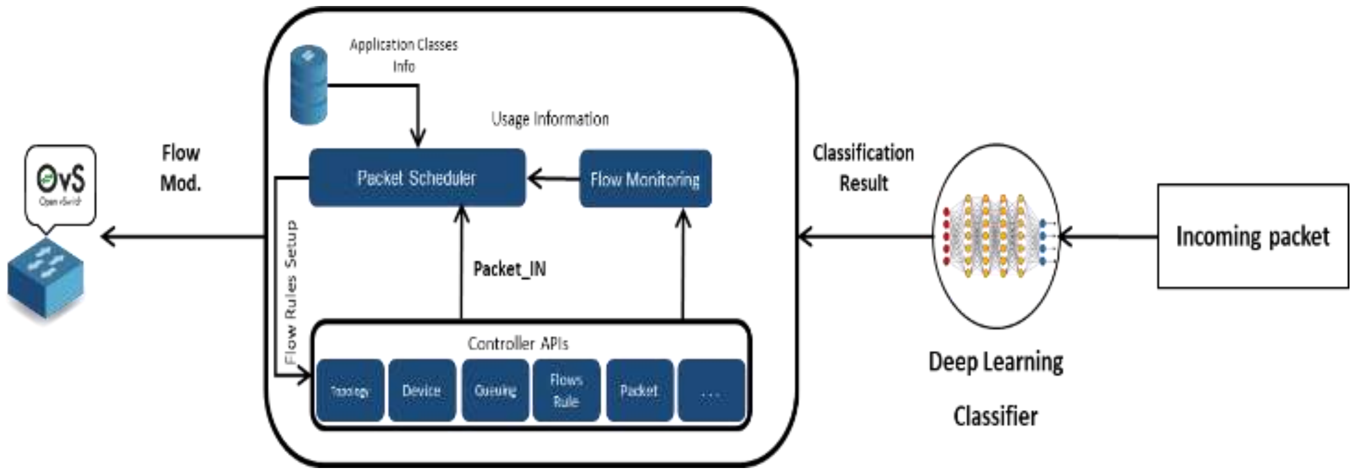

Fig. 1. System Design

K-Nearest Neighbor (KNN) algorithm is one of supervised learning methods, and it is considered as one of the simplest of all machine learning algorithms. KNN works by memorizing the training set, and then try to predict the label of new samples based on the labels of its closest neighbors in the training set. The classification of samples is applied to determine the distance between the unlabeled sample and its neighbors. Such as City-block, Euclidean, Chebyshev and Manhattan are used to measure the distance. Support Vector Machines (SVM) algorithm is also one of supervised learning methods, it is commonly used for classifying objects and pattern recognition. SVM works by mapping the input vector into a high-dimensional feature scope, and the mapping is done by implementing different kernel functions. Kernel function selection, such as linear, polynomial and Radial Based Function, is considered as a crucial job in SVM which can have effect on the accuracy of the classification result. The purpose of Support Vector Machine is finding a separating hyperplane in the feature scope to maximize the margin among different classes [18].

Decision Tree (DT) algorithm is a supervised learning technique that is used to perform classification by a learning tree. DT is consisting of internal decision nodes and terminal leaves. Each node in the tree denotes an attribute, branches denotes the conjunctions of attributes that lead to classifications, and each leaf is a class label. The unlabeled samples are classified according to its feature value with the decision tree nodes. DT is easy to implement and it is capable to achieve high accuracy. Three of most common decision tree algorithms are ID3, C4.5 and CART. The core difference between them is the splitting criteria.

Random Forest (RF) algorithm is a supervised learning technique that is used for classification and regression problems. RF made of many decision trees. RF is randomly choose a subset of the feature scope to construct each decision tree in order to minimize the over-fitting and improve the classification accuracy. RF works by putting the samples to each tree, each tree gives a result which is the tree's vote, then the samples will be classified into the class according to which has the most votes [19].

3.2. System Design

As mentioned before, the deep learning classifier is located at the network's edge to capture and analyze the incoming flow. OpenFlow networks typically run in a reactive mode that means when the packet arrives at the network, the switch looks it up in its flow tables. If there is no match found, the switch forwards the packet to the controller asking how to handle the new arrival packet. The controller has no information about which application generates traffic and how much network resources it needs. Therefore, the first packet of flow will be replicated and directed to the classifier; when the packet arrives at the classifier, it will be classified according to the previously defined classes, then the classified packets will be tagged with the result of the corresponding class through modifying Type of Service (ToS) field in the IP packet header. After that, the packet will be sent out to the controller.

Once the controller receives the packet from the classifier, it decodes ToS in the packet header and matches the value of ToS with rules installed on the controller; after that, packet flows are mapped to a specific queue by enqueueing, and flows will be treated according to that queue's configuration. The decisions taken by the controller will be sent to the switches to install them in their routing tables, so the following packets that belong to the first packet will be handled the same way.

Since OpenFlow does not support queue configuration, as a result, the configuration and setup are done in the OpenFlow switch. Also, the OpenFlow switch allows us to specify a packet scheduler type and prioritize flows. The packet scheduler works on attaching or detaching packets for queues and allows for prioritizing flows by setting different priority values for each queue, enabling flow to be handled differently and guarantee the necessary communication. So, low priority flows are installed first, and then the other flows are installed. The HTB packet scheduler is used to delay packets to meet each queue's desired rate, and it enables to configure minimum and maximum bandwidth capacity. Configuring the bandwidth capacity allows for allocating bandwidth for each queue to guarantee the minimum bandwidth for each application type and avoid bandwidth starvation. After the flows are enqueued to the related queue and handled by the packet scheduler, traffic shaping is used to manage network bandwidth by controlling the rate and the volume of incoming flows to avoid congestion. Applying traffic shaping increases the network's performance, bandwidth availability, and prevents an unexpected increase in bandwidth usage.

The controller looks up the source IP and destination IP to make a routing decision, so the controller from the global view calculates multiple routes between the source and the destination. In general, it selects the route with the least hop counts, though it is possible to select the same route to transmit dense flows, causing congestion. For the routes with the same number of hop counts, port-based algorithms consider each port's capacity to accomplish general load balancing. A load balancing technique is implemented on the switch that works on splitting the workload on the number of physical links to decrease traffic congestions, increasing the throughput, and using the available physical links effectively. It tries to accomplish a general load balancing by considering the capacity of each egress port.

The flow monitoring module monitors the link availability by collecting the delay and available link bandwidth of the physical network periodically. Also, other models can acquire all the monitor information.

3.2.1. OpenFlow Switch

The OpenFlow's current specification does not support queue configuration; thus, queue configuration is being handled by specific OF-CONFIG and Open vSwitch Database Management Protocol (OVSDB) protocols. Even though the OVSDB protocol is already implemented in Open vSwitches, there are no controllers that offer consistent queues management [5].

In the proposed system, we used Open vSwitch as the OpenFlow switch. Open vSwitch is an open-source virtual multilayer switch intended for programmatic network automation, it supports implementing the OpenFlow protocol, and its implementations consist of flow tables with each flow entry having match conditions and associated actions. Open vSwitch connects to the controller through a secure channel and uses OpenFlow protocol to control network flows. Open vSwitch supports a wide range of tools, such as managing queuing disciplines, switch port queues, port mirroring rules, and other tools [13].

3.2.2. Queuing

Queuing is used to provide QoS by managing the queues that hold packets while waiting for their turn to be transmitted. When a network device receives packets, it makes a forwarding decision and sends the packets to the outgoing interface. However, when the outgoing interface is busy, it keeps the outgoing packets in queues, waiting for the outgoing interface to be available [14].

The queuing system requires a scheduler to decide which packet will be transmitted next when the interface becomes available. Also, the scheduler can perform prioritization, giving a certain level of priority to a queue. Each queue can have a different priority value in a multi-queue system that enables that queue to handle packets over other queues. Hierarchical Token Bucket (HTB) is used to manage our queuing discipline, enabling network admins to assign a particular priority to each set of data packets based on what type of transmission it is.

3.2.3. Quality of Service Module

The Quality of Service (QoS) module in the system design consists of three main components: Enqueueing, Packet Scheduler, and Traffic Shaping. The Enqueueing component is responsible for administering the OpenFlow flow table messages and mapping flows to corresponding queues; it is located inside the controller. Whereas, The Packet Scheduler is responsible for managing the packets in queues, and Traffic Shaping is in charge of manipulating the bandwidth volume in queues, both of these components are located inside the switch.

- Queue Structure

The OpenFlow based switches can have one or more queues attached to a particular output port, and those queues can use packet scheduler to schedule existing packets the datapath on that output port. Each queue is uniquely identified by queue ID and port number. Two queues may have the same queue ID. Packets are directed to one of the queues based on the packet output port and queue ID; it is done using the Output and Set Queue actions. Packet flows are mapped to a specific queue by enqueueing, and flows will be treated according to that queue's configuration.

- Enqueueing

It is in charge of operating OFPT FLOW MOD messages of the OpenFlow Protocol. This message works on modifying the state of the flow table. Each packet has action, counters, and header fields for matching flow packets. The enqueueing maps packet flows to queues using SKB PRIORITY of kernel's data structure named SK BUFF. The configuration is completed by using the SO PRIORITY option of the packet.

- Traffic Shaping and Packet Schedulers

These two components are responsible for running QoS messages received from the control plane by allocating the bandwidth volume in queues and attaching or detaching packet schedulers for these queues. Furthermore, these components manipulate OFPT QOS QUEUEING DISCIPLINE messages representing the Quality of Services messages in the OpenFlow protocol. HTB packet scheduler is used to delay packets to meet each queue's desired rate, and it enables to configure minimum and maximum bandwidth capacity.

4. Experimental Evaluation

4.1. Evaluation of the Deep Learning Models

The Scikit-learn library is used to train the machine learning model, and the PyTorch library is used to train the deep learning models. The duration of the flows of 5, 10, 15, and 30 seconds was taken into consideration to conduct our experiment. We have also examined different flow timeout to demonstrate the flow timeout effect on the final results

The performance of deep learning models is compared with several machine learning models, such as K-Nearest Neighbors, Support Vector Machine, Decision Tree, and Random Forest. To improve the models' performance, the datasets used to train machine learning models have been standardized, and the datasets used to train deep learning models have been normalized. Since the classification's overall accuracy does not give a good insight into the classifier performance; thus, precision, recall, and F-score metrics are used to assess the models' performance. We used 5-fold cross-validation during the evaluation of deep learning and machine learning models. Cross-validation is used together with regularization to aid in tuning the hyperparameters. Cross-validation also offer the ability to test the model on unseen data while maintaining a test set that will be used for testing at the end. At the time of training, we did an exhaustive search over specified parameter values for both deep learning and machine learning models.

Accuracy is the proportion of correct classifications.

$$Accuracy = TP + TN \,/\, (TP + FP + FN + TN)$$

Fig. 2. shows the accuracy of both ML and DL models.

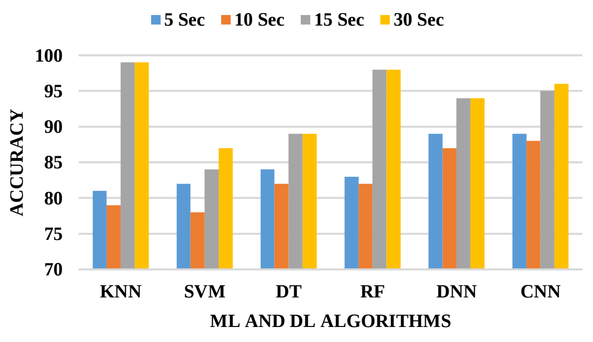

Fig. 2. The accuracy of ML and DL models.

Precision is the number of true positives divided by the total number of true positives and false positives predicted by the model. The precision can capture features that determine the positiveness of a sample to avoid misclassification as negative.

$Precision = TP/(TP + FP)$

Fig. 3. shows the precision of both ML and DL models.

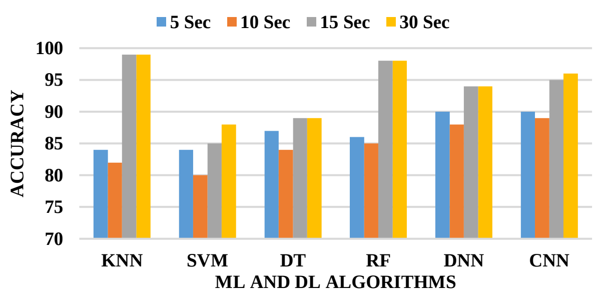

Fig. 3. The Precision of ML and DL models.

The recall, also known as sensitivity, can identify true positive samples between all the potential positives, including the false negatives.

$Recall = TP/(TP + FN)$

Fig. 4. shows the recall of both ML and DL models.

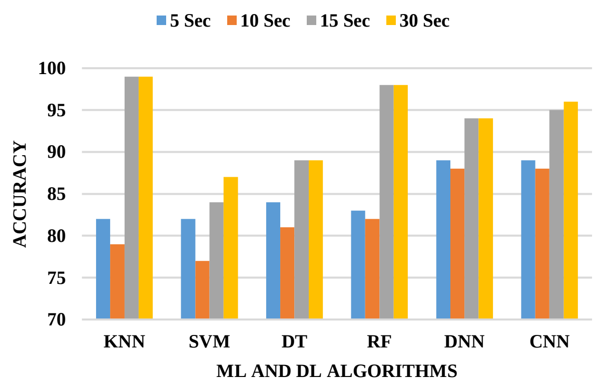

Fig. 4. The recall of ML and DL models.

F-score is a combination of precision and recall are in a single criterion, and it is a measure of a test's accuracy.

$F1 = 2 \times (Precision \times Recall)/(Precision + Recall)$

Fig. 5. shows the F-score of both ML and DL models

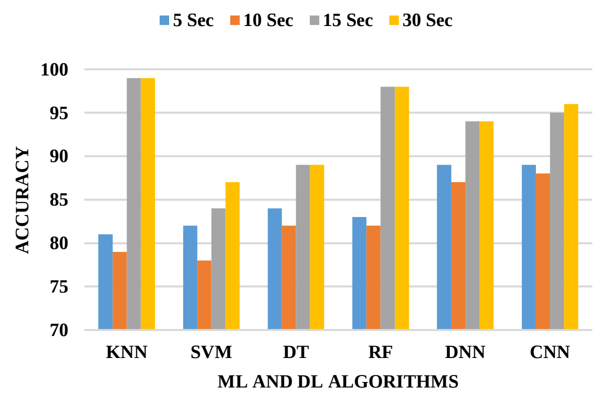

Fig. 5. The F-score of ML and DL models.

It is worth noting that the advantages of implementing DL over ML models are the following: scale-up effectively compared to other ML algorithms, provides better performance, and needs no feature engineering.

4.2. Evaluation of the Overall System

In this experiment, we used Mininet, which is a network emulator, as a testbed to implement our system on an SDN topology, and we used the POX controller as the SDN controller [16] [17]. During the evaluation stage, we used Open vSwitch as an OpenFlow switch.

To evaluate our system, we used three different scenarios: the first scenario is the baseline where nothing is implemented, the second scenario involves the implementation of queues only, and the third scenario involves implementing queues and load balancing together. The queue rules and load balancing are installed on the OpenFlow switch, and the rules of the packet scheduler are implemented on the controller.

The link's bandwidth between connected devices is set to 100 MB each, and the delay is set from 5ms to 30ms.

Table 3. shows the bandwidth allocation for each group of applications. A TCP connection is set between the server and the hosts during the experimenting stage. Fig. 6. Shows the topology used for the evaluation. We have three hosts connected to a server with each host using a different application type.

Table 3. Application Bandwidth Requirements

Class	*Application*	*Bandwidth*
Priority	Real-time traffic	10 MB
High	Audio-Streaming, Browsing, E-mail	35 MB
Moderate	File Transfer, P2P, Video-Streaming	55 MB

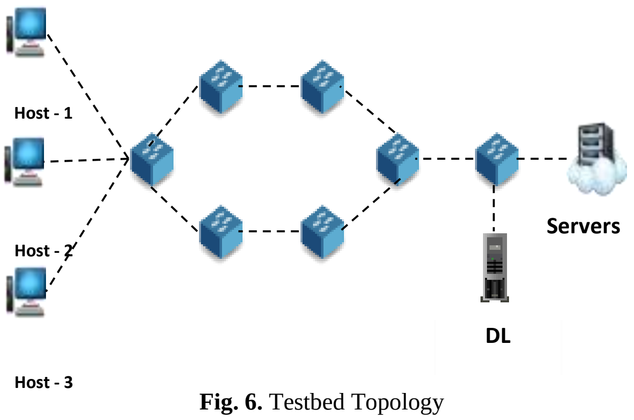

Fig. 6. Testbed Topology

In the first scenario, no rules or policies were implemented to monitor the regular rate of transmitting data between hosts and servers. At the congestions time, the hosts suffer from bandwidth starvation, causing data loss and decreasing in receiving and transmitting rate between data that create a chaotic environment. This is shown in Fig. 7.

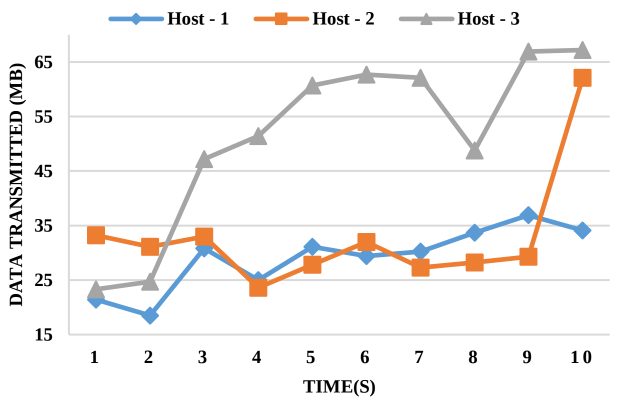

Fig. 7. Data transmission rate between hosts and servers for the first scenario

In the second scenario, rules of queues are installed on the OpenFlow switch, and then traffic shaping is applied to the traffic flow. Applying traffic shaping increased the performance of the network and the availability of bandwidth. The regulation of bandwidth usage helps in guaranteeing resources for each type of application. This is presented in Fig. 8

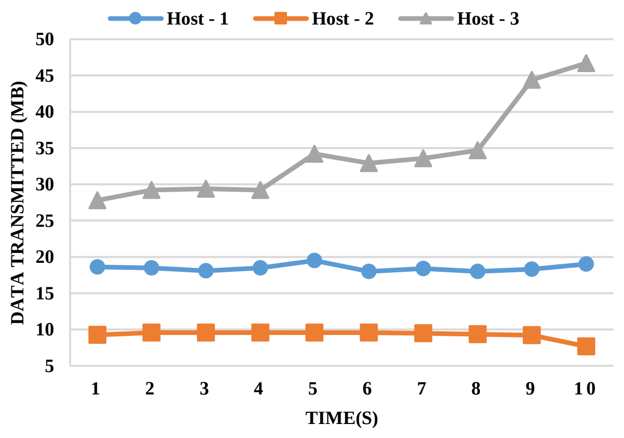

Fig. 8. Data transmission rate between hosts and servers for the second scenario

In the last scenario, the same rules of queues are installed on the OpenFlow switch, and traffic shaping is applied, in addition to installing a load balancing technique on the OpenFlow switch too. This is displayed in Fig. 9.

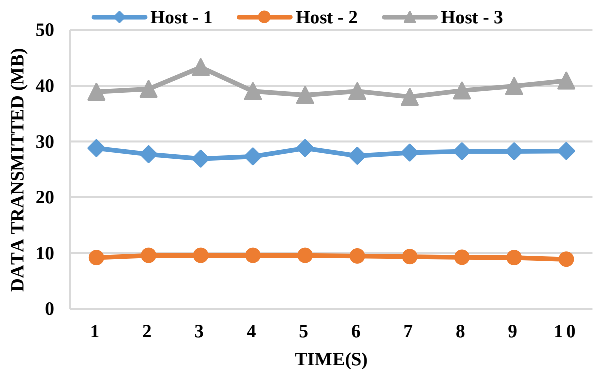

Fig. 9. Data transmission rate between hosts and servers for the third scenario

5. Conclusion

In this paper, we propose a traffic engineering system for SDN based on deep learning models to improve bandwidth allocation among various applications by classifying the network traffic using time-based characteristics of flows and applying QoS. The system distributes traffic to multiple queues with different priorities to improve the QoS of the traffic flows. To increase the network's performance and avoid traffic congestion, we implement a technique to achieve general load balancing. The experimental result of the traffic captured at 5s and 10s flow timeout showed CNN and DNN were are to achieve above %88 accuracy more than machine learning models, while KNN and RF are able to achieve more than %98 accuracy of traffic captured in 15s and 30s timeout better than CNN and DNN that are able to achieve more than %94 accuracy. The advantages of implementing DL over ML are the ability to scale-up effectively compared to other ML algorithms, provides better performance, and needs no feature engineering. The evaluation of the overall system showed applying traffic shaping to the identified flow increases the network's performance and bandwidth availability, it also showed that we can guarantee bandwidth for each type of application, and implementing the load balancing technique showed an increase in the throughput of each queue.

As future work, we plan to test the system on larger network topologies and study the effects of more hosts being involved in the data transfer. Another interesting research idea is studying the impact of increased traffic volume per host and more variety in the applications generating the traffic.

References

[1] Mendiola, A., Astorga, J., Jacob, E., & Higuero, M. (2016). A survey on the contributions of software-defined Networking to traffic engineering. IEEE Communications Surveys & Tutorials, 19(2), 918-953.

[2] Robertazzi T.G., Shi L. (2020). Machine Learning in Networking. In: Networking and Computation. Springer, Cham. https://doi.org/10.1007/978-3-030-36704-6_7.

[3] Yan, J., & Yuan, J. (2018, August). A survey of traffic classification in software defined networks. In 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN) (pp. 200-206). IEEE.

[4] Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. IEEE communications magazine, 57(5), 76-81.

[5] Karakus, M., & Durresi, A. (2017). Quality of service (QoS) in software defined Networking (SDN): A survey. Journal of Network and Computer Applications, 80, 200-218.

[6] Jeong, S., Lee, D., Hyun, J., Li, J., & Hong, J. W. K. (2017, September). Application-aware traffic engineering in software-defined network. In 2017 19th Asia-Pacific Network Operations and Management Symposium (APNOMS) (pp. 315-318). IEEE.

[7] Yan, J., & Yuan, J. (2018, August). A survey of traffic classification in software defined networks. In 2018 1st IEEE International Conference on Hot Information-Centric Networking (HotICN) (pp. 200-206). IEEE.

[8] Amaral, P., Dinis, J., Pinto, P., Bernardo, L., Tavares, J., & Mamede, H. S. (2016, November). Machine learning in software defined networks: Data collection and traffic classification. In 2016 IEEE 24th International Conference on Network Protocols (ICNP) (pp. 1-5). IEEE.

[9] Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., & Ghorbani, A. A. (2017, February). Characterization of Tor Traffic using Time based Features. In ICISSP (pp. 253-262).

[10] Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. IEEE communications magazine, 57(5), 76-81.

[11] Choubey, R. N., Amar, L., Khare, S., & Venkanna, U. (2019, December). Internet Traffic Classifier Using Artificial Neural Network and 1D-CNN. In 2019 International Conference on Information Technology (ICIT) (pp. 291-296). IEEE.

[12] Xu, J., Wang, J., Qi, Q., Sun, H., & He, B. (2018, September). Deep neural networks for application awareness in SDN-based network. In 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). IEEE.

[13] The Linux Foundation. (n.d.) Open vSwitch . Retrieved June 19, 2020, from http://openvswitch.org/

[14] Thazin, N., Nwe, K. M., & Ishibashi, Y. (2019, February). Resource Allocation Scheme for SDN-Based Cloud Data Center Network. Seventeenth International Conference on Computer Applications (ICCA 2019).

[15] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

[16] Kaur, S., Singh, J., & Ghumman, N. S. (2014, August). Network programmability using POX controller. In ICCCS International Conference on Communication, Computing & Systems, IEEE (Vol. 138, p. 70).

[17] Mininet. Retrieved May 15, 2020 from http://www.mininet.org.

[18] Shalev-Shwartz, S., & Ben-David, S. (2014). Understanding machine learning: From theory to algorithms. Cambridge university press.

[19] Xie, J., Yu, F. R., Huang, T., Xie, R., Liu, J., Wang, C., & Liu, Y. (2018). A survey of machine learning techniques applied to software defined networking (SDN): Research issues and challenges. IEEE Communications Surveys & Tutorials, 21(1), 393-430.

[20] Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2019). A survey of deep learning and its applications: A new paradigm to machine learning. Archives of Computational Methods in Engineering, 1-22.