The Journal of Supercomputing (2021) 77:8038–8062
https://doi.org/10.1007/s11227-020-03613-3

Traffic classification for efficient load balancing in server cluster using deep learning technique

V. Punitha[1] · **C. Mala**[2]

Accepted: 29 December 2020 / Published online: 12 January 2021

Abstract
Extensive use of multimedia services and Internet Data Center applications demand distributed deployment of these applications. It is implemented using edge computing with server clusters. To increase the availability of the services, applications are deployed redundantly in server clusters. In this situation, an efficient server allocation strategy is essential to improve execution fairness in server cluster. Categorizing the incoming traffic at server cluster is desired for the improvement of QoS. The traditional traffic classification models categorize the incoming traffic according to their applications' type. They are ineffective in selection of suitable server, as they do not consider the characteristics of the server. Hence this paper proposes a classifier to assist the dispatcher to distribute the requests to appropriate server in server cluster. The proposed deep learning classification model based on incoming traffic characteristics and server status is reinforced with extended labelling using correlation based approach. The experimental results of the proposed classifier have shown considerable performance enhancement in terms of classification measures and waiting time of the requests compared to existing machine learning models.

Keywords Network traffic classification · Server allocation strategy · Server cluster · Deep learning technique

1 Introduction

The technological advancements in computing and the availability of increased bandwidth empower the users to request internet services and data center applications from anywhere, and at any time. The application servers are receiving

✉ V. Punitha
vpunitha21@gmail.com

[1] Department of Computer Science and Engineering, Saranathan College of Engineering, Tiruchirappalli, India

[2] Department of Computer Science and Engineering, National Institute of Technology, Tiruchirappalli, India

enormous number of requests due to high demand. As a result, the computational complexity of the data center/cloud server is increasing. To meet the present high demand, the availability of the services is increased. As a consequence, the deployment architecture of internet applications has undergone a huge transformation, i.e., the work of the central cloud server is distrubuted to network edges which improves the availability of the services and reduces the dependency and end to end latency [1]. This technology is called as edge computing. The edge computing is organized with server clusters, data centres and servers connected by high performance network [2].

A server cluster is a collection of autonomous servers networked together to give a single system appearance [3]. The performance of edge computing depends on server cluster. To improve the QoS of server cluster, QoS in resource allocation strategy at server cluster is to be improved. Analysing the incoming traffic at network edge provides valuable findings about applications' demand and users' need. This kind of information is applied for network management operations. It is also used by the dispatcher of the server cluster to design an accurate server assignment policy for the improvement of QoS in server cluster.

Classification of requests at dispatcher according to their applications helps to assign the respective application server for the requests. Many classification mechanisms have been proposed to categorize the network traffic [4–6]. Port-based approach, payload based approach, deep packet inspection approach and statistical signature approach are widely used mechanisms to categorize the network traffic [5, 7]. Recent internet applications use dynamic ports. Hence port based approach cannot be always used to classify the real-time traffic. Mostly, the payload of the traffic is encrypted to preserve users' privacy. So deep analysis in the payload is infeasible. Flow based approaches bring out behavioral characteristics of network traffic better than packet based approaches [5]. Moreover, the existing traffic classification mechanisms classify the traffic based on the applications, especially based on standard and widely used applications [8, 9]. They are applied to select appropriate application server for each request.

Recently, many applications instances are deployed in the same application servers and these applications share common IP address and port numbers [8]. There are many such servers in a server cluster [1, 2]. Furthermore, application instances are also deployed in one or more servers to improve the availability of the services. In this situation, classifying the traffic according to the application type is not adequate, because this classification is ineffective in identifying appropriate application servers. They do not consider present load on the servers. Beyond IP address and port, the server status is very much essential to classify the requests, so that they can be distributed to suitable servers. This emphasizes an automatic categorization of traffic at network edge for the improvement of QoS and it is yet to be discussed. Hence a novel classification model, considering the users' requesting behaviour and properties of server cluster, is necessary to classify the request for efficient server allocation scheme.

2 Literature survey

This literature survey presents few existing methodologies pertaining to the classification of network traffic proposed by various authors. Authors in [10] presented a SVM classification model exclusively for UDP traffic. Here signatures were derived from incoming network traffic using chi-square-like test. Then, the derived signatures were given as input to classification model. Authors tested the performance of the model for different types of application traffic such as VoIP, P2PTV and traditional P2P applications. Flow level statistics were applied to categorize applications of UDP traffic in [11]. A graph was constructed with the bidirectional flow and the components of the graph were identified. Each application was defined as a component in the graph. Authors suggested that P2P applications were distinguished using the ports of the component.

In [12], authors performed classification for real time data using four machine learning methodologies such as C4.5 decision tree, SVM, Naive Bays and Bayes Net. Different types of application and their network flows were analysed. Authors proved that the classification accuracy of C4.5 decision tree methodology was better than others.

In [13], authors presented a SVM based P2P flow identification method. Authors presented a review on identification of P2P traffic using multidimensional flow properties. The properties were selected using discriminatory selection procedure to improve the classification accuracy. But it does not tested on real-time P2P traffic. In [14], authors proposed a classification model with optimized feature selection procedure. They applied correlation analysis and redundancy filtering. The performance of feature selection procedure was evaluated with various supervised learning techniques.

In DAG-SVM, the learning structure was implemented with series of binary classifier organized in DAG structure. Here when the classification error appreared, it spread over to other layers. In the proposed Improved DAG-SVM, the learning structure was defined by two metrics called distance and decision function [8]. The classification errors were rectified in the layer where the errors occurred. Here, two learning structures were practiced to classify the incoming network traffic according to the type of the application.

A novel signature based scheme was proposed to differentiate the applications of the traffic in [15]. Here authors constructed signature from first few bits of a network flow. The signatures were compressed and an automata engine was created to recognise the pattern. But this model classifies the traffic according to the applications. A new traffic classification model was proposed in [16] to classify both UDP and TCP traffic using protocol signature. The proposed model reconstructed itself in constant interval of time using the previous classification results and protocol distribution. Network packet properties were removed and packet document was created using payload and this document model was applied for classification. Encrypted payload data could also be classified using this model, but frequent reconstruction of the model is additional overhead and time consuming.

In [17], a traffic classification model was proposed to eliminate noise in traffic dataset for the improvement of reliability of training dataset. The proposed model extracted statistical characteristics of packet features such as number of packets, mean and standard deviation of packet size and inter packet time. The model discovered noisy traffic by finding the misclassification and redefining those traffic. But multi-tenant application features are not considered.

New traffic classification model called, Automated Packet Payload Content (PPC) was presented in [18]. Signatures were generated according to the applications using static payload. Subsequently, the traffics were classified as text and binary based applications. Authors proved that wrapped applications or applications using random ports could also be identified by PPC.

Usage of supervised algorithms are controlled due to complexity of labelling the unknown data. To overcome this issue, authors in [19] proposed semi supervised framework using flow correlation . Flow label propagation method which recognized the unknown flow using correlation measure and Nearest Cluster based Classifier (NCC) using k-means technique were proposed in [19]. After grouping the flows into clusters, NCC mapped them into different classes according to their applications. Authors tested their compound classification model with Bag-of-Flows (BoF).

In [9], authors proposed a classification model to identify zero-day applications based on flow correlation. The features were selected using optimization technique. The proposed model applied supervised learning approach as well as unsupervised learning approach. A SVM classification model was proposed in [20]. The incoming traffics were classified based on flow features and state information of the servers. Here the execution fairness was improved by the proposed classification model, but the feature vectors are not updated dynamically for each request.

A new resource allocation framework was proposed exclusively for Storm cluster with predefined priority channels in [21]. Authors measured QoS violation of the application and CPU usage. Authors observed the QoS violation in terms of expected latency. But the application streams are classified as only low, medium and high priority classes.

Two models were constructed in [22], MLP and CNN models and four types of traffic were classified using deep learning methodology. Authors proved that performance of CNN model was good for audio traffic; MLP was good for other traffic such as text, image or video. But number of hidden layers and neurons in each hidden layer are very minimum. A new virtualised network function was proposed for SDN [23]. It distinguished the type of the traffic using port number. This function captured the traffic for limited period and computed the features. DNN architecture was implemented, but only inbound traffic features are applied for classification.

A deep learning classification model was proposed for IoT traffic in [24]. Deep architecture was constructed with CNN and Recurrent neural network (RNN). Minimum number of features were selected using the proposed feature selection procedure. But application using dynamic port is not considered for classification.

A reinforcement learning approach was proposed to overcome the unbounded state space issues in [25]. This model-based approach learnt the control policies for queuing network to minimize average queue backlog. The theoretical guarantees were corroborated on server assignment problems.

Authors proposed an integer linear programming model for server allocation under many failure prototypes [26]. Initially, the model was designed with optimum allocation without failures, called start time optimization. Then, when failure happened, it reallocated the user to another server considering the delay. Authors proved that instability during connection failure was eliminated.

Authors proposed a bi-criteria approximation model in [27] to optimize the allocation of jobs to server considering the total cost of the schedule. This model was designed with many exceptional situations such as identical servers, constant number of servers having cost constraints etc. The model also developed approximation systems with linear constraints for a varied classes of load balancing problems.

A continuous time resource assignment problem with fairness reviews was studied in [28]. Considering utility and total time, a new structure was proposed and theoretical results were verified.

Authors in [29], proposed a new server allocation strategy in cloud environment. Here cuckoo optimization algorithm was applied to select a server by avoiding congestion in the network. The selection was obtained by finding the suitability of the respective virtual machines by the cuckoo algorithm. The selected virtual machine accomplished the user's works; the other virtual machines were ignored.

A new load balancing algorithm for fog computing environment was proposed in [30]. Here, to minimize the energy consumption, the tasks were allocated equally to all the nodes in fog layer. When a node was overloaded, the algorithm transferred the task to nearby free node. But compared to this method, nature inspired methodologies produce better results.

In [31] authors proposed a novel procedure for server allocation problem in virtual computing labs. This data driven procedure allocated the servers to dynamic demands, considering system cost and quality of service. The quality of service was measured by finding fraction of clients who gained delayed or immediate access to the server and the system cost by finding available number of flexible and preloaded servers. Singular value decomposition was used to estimate future demand and stationary dependent period by period was used to deal with dynamic system.

Allocation of edge servers was implemented in [32] by considering two types of deployments such as flat and hierarchical deployment. M/M/c queuing systems were modeled for both deployment types. In flat type, if the arrival rates were same, allocation of servers was unbiased across all base stations. Turnaround time was minimized in hierarchical deployment when none of the edge cloud was co-located with base station.

In [33], authors proposed a resource allocation framework in multi-access edge computing environment to minimize the number of migration events instead of maximizing the resource usage, as the migration caused service downtime. The proposed model avoided migration of low priority services by reserving adequate resources for high priority services. Number resources required for high priority services were estimated through learning automata. The method was evaluated using expected admission time metrics.

A novel cloud architecture was proposed in [34] for MMORPG game. Server allocation to the game players was performed using a heuristic algorithm which

minimized server rental/execution cost. The performance of the algorithm was compared with online heuristic algorithms in terms of scalability.

Based on the presented survey, it is observed that the incoming traffics are classified according to their applications using network packet features so that they can be distributed to appropriate application servers. But classification of incoming traffic for the improvement of server utilization requires the following deliberations.

- Packet header information are adequate to categorize the real time traffic based on their applications. But other distinct features of network traffic and servers are essential to categorize the traffic to improve the execution fairness.
- Deep packet inspection approaches are good at classifying network traffic according to the type of the applications. But flow based classification identifies the applications in dynamically changing environment more effectively. Only few classification methods applied server characteristics in classifying incoming requests, but it is more essential for the improvement of execution fairness.
- The applications of the traffic are identified using information theory by few authors. Machine learning approaches are used by many authors. Deep neural network architecture will produce better learning accuracy, as it brings out latent characteristics in network traffic and it is yet to be investigated.

The above assessment emphasizes automatic classification of real-time traffic for the improvement of QoS in server cluster. In this regard, this paper,

- Proposes a model to classify the requests at server cluster in multi-tenant environment where multiple application instances are deployed in the same server.
- Examines the characteristics of the request in terms of network flow features and also appraises the characteristics of the server.
- Proposes a novel method to obtain server properties dynamically using bag of flow.
- Proposes a novel Traffic Classification for Server cluster using Deep Learning approach (TCSDL) to categorize the incoming traffic based on its characteristics and server status.
- Proposes a correlation based approach to integrate unknown network traffic in bag of flow with labelled network traffic.
- Proposes an Extended Learning approach to identify and define the classes for unlabeled unknown traffic when number of labelled traffic is less.

The rest of the paper is organized as follows. The proposed classifier and extended learning approach are elaborated in Sect. 3. The experimental results of the proposed classifier are analysed in Sect. 4 and the proposed work is summarized in Sect. 5.

3 Proposed traffic classification model

Over the recent years, nearly all the commercial organizations are offering the services over the internet. Consequently, the development of internet data center applications and the respective traffic towards network edges are increasing. This in turn demands proficient server cluster at network edge. The proficiency of the server cluster can be improved with an efficient request-server assignment strategy. Effective classification of incoming traffic at dispatcher balances the load among the servers and improves execution fairness [35]. Hence this paper proposes a classification model, Traffic Classification for Server cluster using Deep Learning approach (TCSDL) for improved server allocation in server clusters using deep learning approach.

3.1 Deep learning technique

Many research works have been carried out in network traffic classification using machine learning techniques [5, 6]. Deep learning technique is a special kind of machine learning technique. It uses multiple layers to gradually bring out higher level features and relationships from the input. The Deep Neural Network (DNN) is modelled with artificial neurons, called as perceptrons. The perceptrons are organized in three layers, namely input, hidden and output layers. The structure and relationships among the features are even more refined in each hidden layer. The DNN is constructed with Artificial Neural Network (ANN) or Convolutional Neural Networks (CNN) or Stacked Auto-Encoder (SAE). SAE is mostly used to extract abstract features from raw input using unsupervised learning. CNN is mostly applied for image and video analysis. High dimensional space is required to represent and to bring out latent features of the network traffic, so in this paper, a deep neural network is constructed with ANN.

3.2 Framework of the proposed TCSDL

Usually, the classifier categorizes the incoming network traffic according to their application type based on network packet features. But the proposed TCSDL is designed to classify the incoming traffic to improve the QoS in server cluster, i.e., the incoming requests are classified in such a way that they are always assigned to a least loaded server. For this purpose, the packet features are alone not enough, hence the status of the server cluster is also added into the feature set. The status of the server cluster is to be acquired dynamically. But including a dynamically changing variable into the feature set is complex. Hence the proposed TCSDL constructs Bag of Flows (BoFs), i.e., the network traffic is recorded for a short period of time. The network flows during this period are aggregated and it is called as Bag of Flow (BoF) [19]. The defined time period is called as session window. As it is a short period, the status of the server will not change during this period. Hence the status of the server is obtained for every session window and it is updated in the feature

set. Subsequently the network traffics are labelled based on the feature set, and further TCSDL is trained. The framework of the proposed TCSDL is depicted in Fig. 1.

In traditional supervised learning framework, the network traffic classifier learns the features of labelled training traffic and classifies the incoming real time traffic into predefined classes. This method is degraded when the size of the labelled network traffic is small. To overcome this weakness, this paper proposes a new method to extend the labelled network traffic automatically during training. The proposed method finds the correlation between labelled traffic and unlabelled unknown traffic. Closely associated labelled traffic for any unknown traffic is identified and the corresponding class label is assigned to the unknown traffic. The correlations between traffics are found only within Bag of Flow (BoF), i.e., for a short period of time, as the server cluster will not change its characteristics during this period [19], i.e., when two clients are connected to the server on a specific port (i.e., specific application), the traffic generated by these clients are more likely be identical, as the servers will not change their applications in short span of time [9, 19]. Thus, in this period or in BoF, when the correlation between unknown traffic and labelled traffic is high, then they are more identical. Hence the proposed model obtains the correlation between labelled traffic and unknown traffic in each BoF and extends the labelling.

3.3 Feature learning in proposed TCSDL

Generally, the application of the traffic is identified using the packet features such as destination IP and destination port. For effective classification of requests in multi-tenant architecture, this paper considers the status of the server along with the inbound traffic characteristics.

Before classifying the incoming traffic for effective server allocation, the static requests are to be separated. These requests use static IP address and they do not use external database. So they are processed directly and quickly. It is assumed that the static requests are handled by a dedicated server. So the proposed classifier does not consider static requests for classification. They are discriminated using IP address

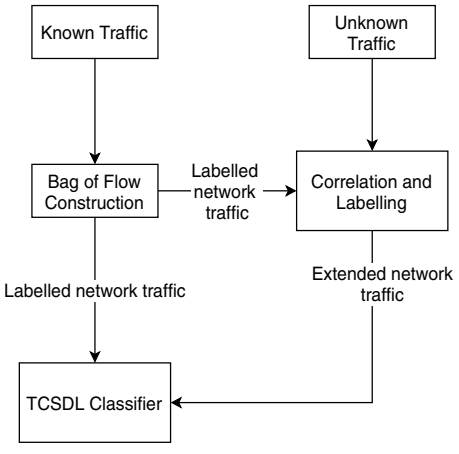

Fig. 1 Framework of the proposed TCSDL

and the size of the response. Remaining requests, other than static requests are taken for analysis. The inbound traffic properties such as source IP, source port and arrival time are obtained from packet header. The upstream and downstream communications concerning to a request are recorded in order and it is named as a network flow. The duration of the network flow defines the service time of a request. The size of the response for each network flow is measured in terms of upstream packets (US) and average outbound packet size (sizeavgOutPktSize). These parameters are listed in Table 1 and represent feature set. The proposed TCSDL classifies the network traffic in such a ways that the requests are always distributed to least loaded server. For this purpose, TCSDL observes the status of the servers in the server cluster in terms of queue size at each server and expected waiting time at each queue during session window. The network flow features and server features that are listed in Table 1 are constructed for each session window.

In existing classification models, authors extracted packet header information and computed the network features for entire traffic [5]. In [9, 19], authors computed 20 flow statistical features from extracted packet header information of the datasets. These classification models are trained and tested using already available datasets. So authors did not consider computational latency. The proposed model is trained and tested using benchmark datasets. So computation of flow features can be implemented in offline. But it is a challenging task when real-time network traffics are tested by the model [36, 37]. Hence the proposed model computes the flow features for each session window. During this period, the packet header properties are extracted and features are computed. Session window is of short duration, so time taken to compute the flow features in this period, is low. The latency of feature learning is further reduced by computing the features in overlapping manner with testing, i.e., when the requests in the previous session window are being classified, the feature learning is performed for the traffic in current session window.

Table 1 The features learnt by TCSDL to classify the network traffic

Feature	Description
srcIP	Source IP address
srcPort	Source port
destIP	Destination IP address
destPort	Destination port
arrivalTime	Arrival time of the request
flowDur	Flow duration
US	Upstream packets in a flow; number of packets communicated from server to client in a flow
avgOutPktSize	Average packet size of the messages communicated from server to client in a flow
waitTime	Expected waiting time at each server queue
qSize	Size of the queue at each server
avgResTime	Average response time of each server

3.4 Proposed traffic classification for server cluster using deep learning approach (TCSDL)

There are three major stages in training the proposed classification model; they are computation of network flow features, construction of bag of flow and updating server features in feature set. The phases of the proposed TCSDL are portrayed in Fig. 2. After removing error packets from the captured traffic, network flows are constructed. For each flow, flow features such as US, avgOutPktSize and flowDur are computed. Then, feature set is constructed with packet and flow features. Then, server status in terms of average response time, queue size and queue waiting time is obtained for each session window and updated in the feature set. Then, the least loaded server is selected according to the request characteristics and server status. Following this, the feature set is labelled. Finally, the proposed model learns the features and classifies the request according to the requested application and server status.

The objective of this classification is, when a new request is received, the proposed classifier defines the class, it is despatched to the server based on the class. In recent times, the application instances are deployed redundantly in many servers in server cluster to improve the QoS. So the request can be executed at any server. In this context, the objective of the classification is to improve the performance of the server. Hence the proposed TCSDL is based on server status, i.e., for example if the server 1 is least loaded, then the request is classified as class 1. It means that the requests which are classified as class 1 are forwarded to the server 1, the requests which are classified as class 2 are forwarded to server 2 and so on. So the requests which are to be executed by a particular server are classified into the same class. The server cluster has more than two servers, so the problem is devised as multiclass classification problem.

The classification scheme proposed in TCSDL to improve the QoS of the server cluster is presented in Algorithm 1. Let $\{T_1, T_2, \ldots, T_p\}$ be the incoming traffic and p be the size of the incoming traffic. Bag of flow is constructed for each session window. Let q be the number of bag of flows. The server features (SF_i) are extracted for each session window. In each bag of flow, the network flows are constructed and packet features are extracted. Then, the flow features (FF_i) are computed. The flow features FF_i and server features SF_i are updated into the feature vector (FV). Here FV_i defines the feature vector for ith flow. Now the TCSDL is trained to categorize the network flow based on flow features and server features. If application is deployed in many servers say $\{S_1, S_2, \ldots, S_l\}$, where $l < n$, then least loaded server Sk is selected using waitTime, qSize, avgResTime. Subsequently the request (F_i) is classified as class C_k. So that F_i is forwarded to the server S_k. The waiting time and response time are computed using the flow duration of the requests in previous BoF.

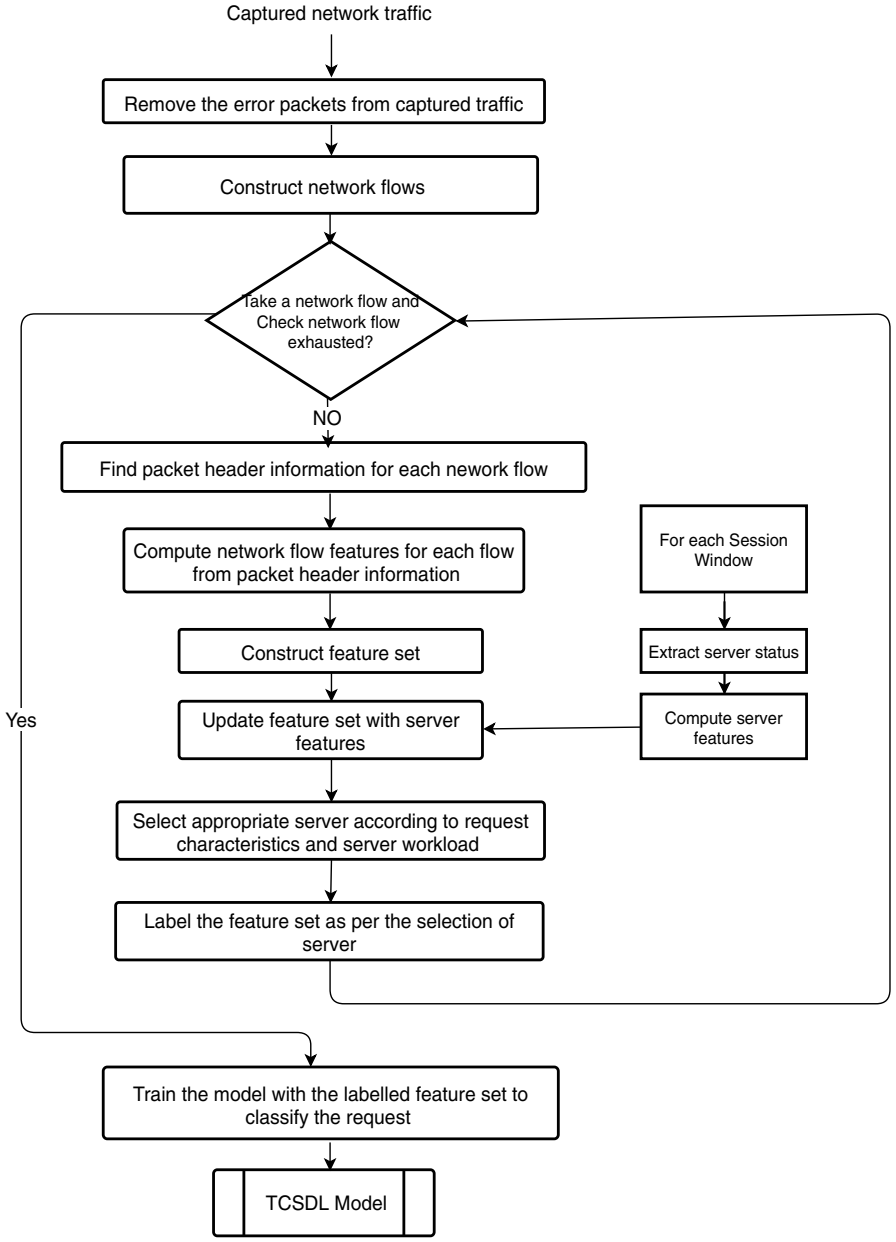

Fig. 2 Flow diagram of the proposed TCSDL

Algorithm 1: Traffic Classification in TCSDL

Description: To classify in the incoming traffic to improve the QoS of the server cluster
Input: Incoming network traffic $\{T_1, T_2, ...T_p\}$
Output: Classified network traffic

1. Construct bag of flow $\{B_1, B_2, ...B_q\}$;
2. **for** *each bag of flow,* $B_i \epsilon \{B_1, B_2, ...B_q\}$ **do**
   a. Obtain server features $SF_i : waitTime, qSize, avgResTime$;
   b. Construct network flow $\{B_1, B_2, ...B_q\}$ from incoming traffic;
   c. **for** *each* $F_i \epsilon \{F_1, F_2, ...F_r\}$ **do**
      i. Extract packet features for F_j;
      ii. Compute flow features $FF_j : flowDur, US, avgOutPktSize$;
      iii. Update FV_j with SF_i and FF_j ;
      iv. Select least loaded server $S_i \epsilon \{S_1, S_2, ...S_l\}$ based on FVj Declare the class of F_i as C_k ;

3.5 Extended labelling in proposed TCSDL

The learning accuracy of the supervised learning depends on training dataset. Labelling the entire traffic or unknown traffic is a challenging task. Hence in this paper, the proposed TCSDL is initially trained with the available training traffic. Then, the labelling is extended further with unknown traffic.

The proposed TCSDL model combines both labelled and unlabelled traffic and constructs a Bag of Flow (BoF). Now the BoF consists of known labelled traffic along with unknown unlabelled traffic. Then correlation among the labelled and unlabelled traffic in a BoF is discovered using Spearman correlation. The Spearman rank correlation measures the degree of relationship between 2 data or variables [38]. The distribution of the data is not considered in Spearman correlation examination. Hence this paper applies Spearman correlation examination to identify the association between unlabelled network traffic and the predefined labelled network traffic. The labelled network flow is described in terms of source and destination address, protocol, service time, upstream packets and average packet size. The packet features of unlabelled traffic is extracted and examined with labelled predefined traffic using Spearman correlation examination. The extended labelling process proposed in TCSDL is described in Algorithm 2.

Algorithm 2: Extended Labelling in TCSDL

Description: To generate labelled traffic for the proposed TCSDL classifier from unlabelled unknown traffic

Input: Predefined labelled network traffic $\{L_1, L_2, ...L_p\}$, Unlabelled unknown traffic $\{U_1, U_2, ...U_q\}$

Output: Extended labelled network traffic

1. Combine labelled and unlabelled traffic, $\{T_1, T_2, ...T_p + q\} = \{L_1, L_2, ...L_p\}U\{U_1, U_2, ...U_q\}$;
2. Construct bag of flow $\{B_1, B_2, ...B_v\}$ from $\{T_1, T_2, ...T_p + q\}$;
3. **for** *each bag of flow,* $B_i \epsilon \{B_1, B_2, ...B_v\}$ **do**
   a. Construct network flow $\{F_1, F_2, ...F_r\}$ from incoming traffic ;
   b. **for** *each* $F_i \epsilon \{F_1, F_2, ...F_r\}$ **do**
      i. Extract packet features for F_j ;
      ii. Compute flow features FF_j for labelled traffic ;
      iii. **for** *each unlabelled flow* **do**
         Find Spearman correlation rank ρ_k with labelled FF_j;
      iv. Define the class for unlabelled flow as $C_k = max_k(\rho_k)$;

Let $\{L_1, L_2, \ldots, L_p\}$ and $\{U_1, U_2, \ldots, U_q\}$ be labelled and unlabelled network traffic, respectively. The extended labelling algorithm combines both the traffics and generate BoFs. In each BoF, flow features for labelled flows (FF_j) are computed. Then, each unknown flow in BoF is examined with labelled flow using Spearman correlation and the correlation rank (ρ_k) is computed. The rank indicates degree of relationship between two traffic. It is in the range of -1 to $+1$ [38]. If the rank is $+1$, it means that the unlabelled traffic is same as the labelled traffic. If it is -1, then both traffics are different. The closely associated labelled traffic for each unknown traffic in BoF is discovered by finding maximum rank value among the ranks generated for each labelled traffic. Then the respective class is assigned to the unknown traffic. Thus the proposed TCSDL model extends the labelling.

4 Result and performance analysis

This section presents the simulated environment of the proposed system and elaborates its performance assessment. The proposed deep architecture is developed using Google Colaboratory with Tensorflow, Keras and PyTorch. The proposed model is evaluated with machine learning models and existing static method. Matlab tools are used to develop the machine learning models. Wireshark is used to derive the packet parameters from the network traffic [39]. The proposed model is examined with benchmark datasets captured from public repository [40, 41]. The statistics of the datasets are presented in Tables 2, 3, Table 4.

Assumption: The server cluster has heterogeneous servers and the service time of each server is distinct. They are configured with multiple applications.

Traditionally, the incoming network traffics at server cluster are classified according to the applications' type and forwarded to the respective application servers. The

Table 2 Network traffic data from benchmark dataset

Dataset	No. of packets	Time span (sec)	Average packets per second	Average packet size (bytes)
WIDE dataset	30,00,000	6262.78	234.6	692
UNB dataset	35,00,000	7932.72	252.1	707

Table 3 WIDE dataset

Protocol	No. of packets	Percentage of packets	No. of flow	Bytes
TCP	14,59,047	48.63	87,218	47,84,47,419
UDP	10,50,991	35.03	68,090	19,56,15,349
Others	4,89,962	16.33	29,511	1,83,65,628

Table 4 UNB dataset

Protocol	No. of packets	Percentage of packets	No. of flow	Bytes
TCP	16,88,747	48.25	80,284	98,90,28,260
UDP	12,50,072	35.72	60,452	37,20,07,928
Others	5,61,181	16.03	21,957	5,24,73,042

applications of the traffics are identified using IP address and port numbers. But in multi-tenant architecture, multiple application instances are deployed in the same server and also to increase the availability, such multiple servers are organized in distributed places. Hence classifying the incoming traffic based on IP address and port number become ineffective. As the server supports multiple applications, classifying the request based on their applications alone doesn't be much effective. Hence server status is also considered while classifying the requests at server cluster.

Machine learning techniques are pre-eminent in classifying network traffic using packet features [5]. Classifying the requests using unsupervised learning techniques is challenging in multi-dimensional space; dimensionality reduction and optimal selection of 'k' is essential for better classification accuracy. Additionally, evaluating the clustered traffic is also challenging [42, 43]. Hence, supervised learning models such as SVM with different kernels, Decision Tree, Naive Bayes and K-Nearest Neighbors (KNN) are examined in this paper. Apart from this, in complex and high dimensional data, bringing out latent characteristics in the data gives better classification accuracy [44, 45]. Hence in this paper, the server status and the flow features are labelled and so that the proposed deep learning approach can be evaluated using the supervised learning models.

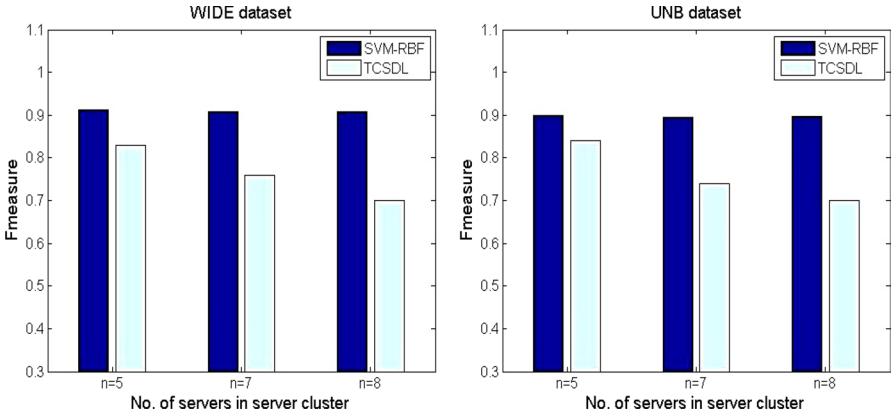

Fig. 3 No. of servers in server cluster versus F measure

4.1 Performance of the proposed TCSDL model

The performance of the proposed TCSDL is measured using classification metrics such as precision and recall. Precision defines the ratio of number of network traffic correctly classified in a given class to the total number of network traffic predicted in the same class. Recall defines the ratio of number of network traffic correctly classified in a given class to the number of network traffic labelled in the same class [5]. In this paper, initially server cluster has only 3 heterogeneous servers to make the prediction rate analysis and average waiting time analysis simple. The classification metrics are computed for all the three class independently and presented in Tables 5 and 6. Precision and recall values in linear and polynomial SVM are lesser among other machine learning models. KNN identifies the similar traffics using nearest neighbour. The server characteristics are uniform in each session window, so it categorizes the traffic more perfectly. Hence precision and recall are greater than linear and polynomial SVM models. Decision tree model is designed with sequence of decisions on the significant features. Waiting time and queue length are used as most

Table 5 Performance of the proposed TCSDL model for WIDE dataset

Learning models	Precision			Recall		
	Class 1	Class 2	Class 3	Class 1	Class 2	Class 3
KNN	0.8188	0.7996	0.7759	0.8395	0.8011	0.7968
Naïve bayes	0.8498	0.8591	0.7989	0.8081	0.8235	0.7438
Decision tree	0.85	0.8027	0.803	0.8017	0.8198	0.8486
Linear SVM	0.7863	0.7375	0.6706	0.7368	0.74	0.7281
SVM-RBF	0.881	0.8674	0.8296	0.8876	0.8285	0.862
Polynomial SVM	0.7933	0.7366	0.7759	0.7914	0.7811	0.7668
DL	0.9297	0.9202	0.9361	0.937	0.9162	0.9302

Table 6 Performance of the proposed TCSDL model for UNB dataset

Learning models	Precision			Recall		
	Class 1	Class 2	Class 3	Class 1	Class 2	Class 3
KNN	0.7541	0.7626	0.7345	0.7786	0.7099	0.7593
Naïve Bayes	0.8486	0.86	0.8174	0.8621	0.801	0.8625
Decision Tree	0.8271	0.8153	0.7715	0.8053	0.8034	0.8109
Linear SVM	0.7425	0.6892	0.7074	0.7462	0.7099	0.6794
SVM-RBF	0.8984	0.9174	0.8297	0.9011	0.8473	0.8969
Polynomial SVM	0.7918	0.8085	0.7869	0.8177	0.8027	0.7593
DL	0.9508	0.9469	0.9287	0.9401	0.9478	0.9417

significant features for classification of the request. Hence the classification performance in decision tree model is also good. Among the machine learning models SVM-RBF model produces best classification results. The proposed multiclass classification model requires more than one hyperplane in multidimensional space. The hyperplanes generated in SVM-RBF classifies each request perfectly. But the latent characteristics between similar traffic are brought out effectively by the proposed DNN architecture.

As SVM-RBF model produces best results among machine learning models, the proposed TCSDL model is further compared against SVM-RBF model with different server clusters which are having 5, 7 and 8 heterogeneous servers respectively. The classification metrics such as precision, recall and F-measure values are computed for each class and the average F-measure values are obtained and plotted in Fig. 3. One Verses One multiclass classification is used in SVM-RBF model. So it generates 10, 21 and 28 hyperplanes to classify 5, 7 and 8 classes, respectively [20]. These hyperplanes categorize the request in multi-dimensional space and it is observed that F-measure values are above 0.7. It is also observed from Fig. 3 that the increase in number of classes does not degrade performance much in TCSDL model. Multiple hyperplanes in SVM-RBF do not discriminate the classes perfectly as in DL. In SVM for better prediction, it is required to calculate optimal arrangement of hyperplanes for splitting the space [46]. Whereas, the DNN architecture in the proposed TCSDL identifies the similarities among feature sets more effectively and predicts the classes flawlessly. This improves the prediction rate and hence the classification performance of TCSDL is better than SVM-RBF model for both datasets.

Along with classification metrics, implementation efforts are also considered for comparison of methods presented in Tables 5 and 6. The implementation has three major phases namely, feature learning, training the model and testing. Same features are applied in all classification models and hence the feature learning phase and computational complexity of this phase are same for all the models. Presently, feature learning is implemented in overlapping fashion. The computational complexity for feature learning could be further reduced using parallel execution of tasks and automatic feature learning [47] which will be examined in the future work.

So comparison among classification models in terms of computation complexity involves training and testing time. The training and testing time taken by the models are measured for the server cluster with three servers and without considering extended dataset. They are plotted in Fig. 4. Among the machine learning models taken for analysis, decision tree and SVM-RBF models produce best performance. Hence training and testing time taken by these models are compared with TCSDL. It is observed from Fig. 4 that time taken for training the decision tree model is minimum among the models taken for analysis. If the training dataset is large, the training time of SVM model will be long [48]. Additional effort is needed to minimize the size of the training dataset using optimization technique which considers only the representative data and minimizes other data in training dataset [48]. As SVM-RBF does not apply any optimization technique in this paper, training time taken by SVM-RBF is longer than decision tree. DNN hidden layers in the proposed TCSDL has multiple nodes, hence time taken for training TCSDL is higher than decision tree and SVM-RBF models. Comparatively, testing time is less. Time taken for training and testing are reduced with effective organization of multiple premium GPUs and cooling system. Above all, prediction accuracy of TCSDL is higher than decision tree and SVM-RBF models and moreover training the model is a one time work. Considering these factors, this paper proposes that DL architecture is ideal for classification of request for effective server utilization.

The proposed TCSDL is compared with existing static model, SCEW [20] in terms of accuracy and f-measure. Accuracy and F-measure are computed for each model and they are plotted in Figs. 5 and 6. Existing SCEW method derived the status of the server, but it is not updated frequently. With this static server features and incoming flow features, feature vector was constructed. Thus, the predicted classes for the incoming requests are not accurate and hence the accuracy and F-measure decreased in Figs. 5 and 6. Whereas the proposed model observes the status of the servers dynamically during each session window and the feature vectors are updated correspondingly. The proposed machine learning (ML) models classifies the incoming requests based on the network flow features and dynamically derived server status. Hence in Figs. 5 and 6, the learning accuracy and F-measure are greater than existing SCEW model. The proposed TCSDL brings out the relationship between

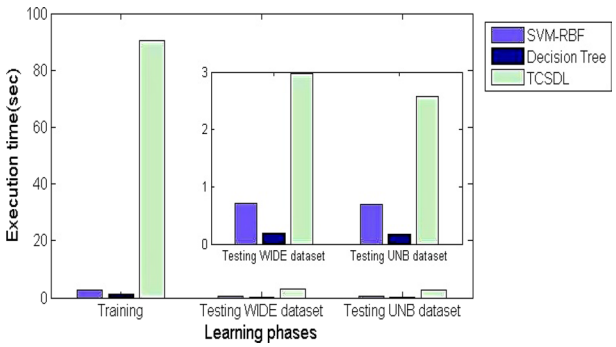

Fig. 4 Learning phases versus Execution time(sec)

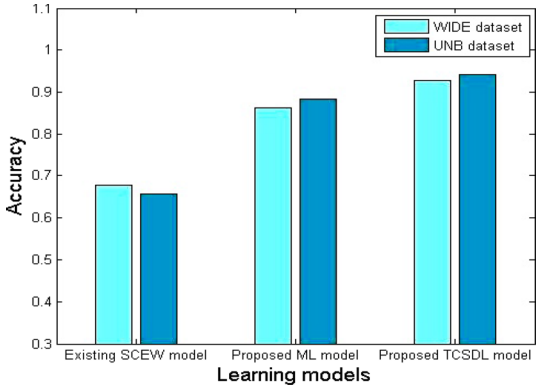

Fig. 5 Learning models versus Accuracy

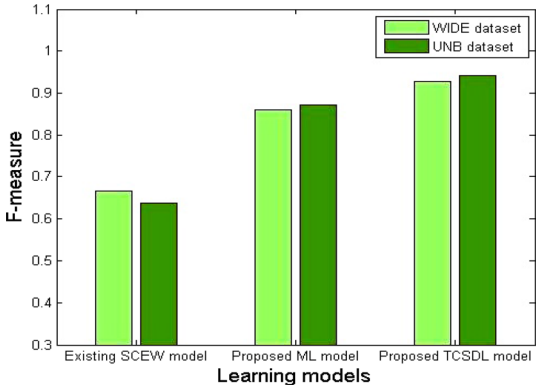

Fig. 6 Learning models versus F-measure

features efficiently in each BoF. Hence the learning accuracy of the proposed TCSDL is greater than existing SCEW and machine learning models in Figs. 5 and 6.

4.2 Prediction accuracy and analysis

The number of requests classified by the proposed TCSDL model is computed for every 2000 arrivals. The prediction rate of request is computed and plotted in Fig. 7. The prediction rate is defined as the number of requests correctly classified divided by total number of requests received. The precision and recall values of each category are above 0.9 for both datasets. Hence the prediction rate in Fig. 7 is always above 0.9. It implies that the proposed classification model proficiently classifies the request considering the load balancing among servers.

Now, the efficiency of proposed model in predicting each type of request is examined. The proposed TCSDL is designed with three servers to make the analysis simple. So it classifies the incoming requests into three categories. The number of requests classified in each category is computed for every 2000 arrivals for both datasets. They are plotted in Fig. 8. The servers in server cluster are heterogeneous.

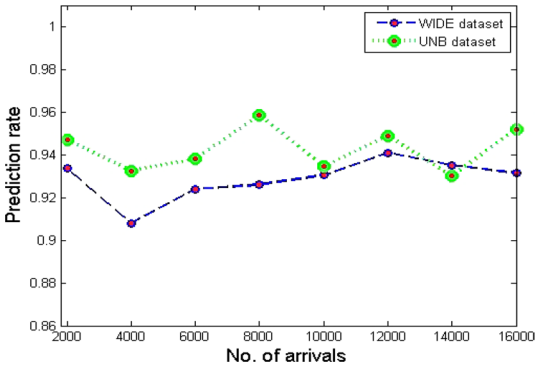

Fig. 7 Number of arrivals versus Prediction rate of request

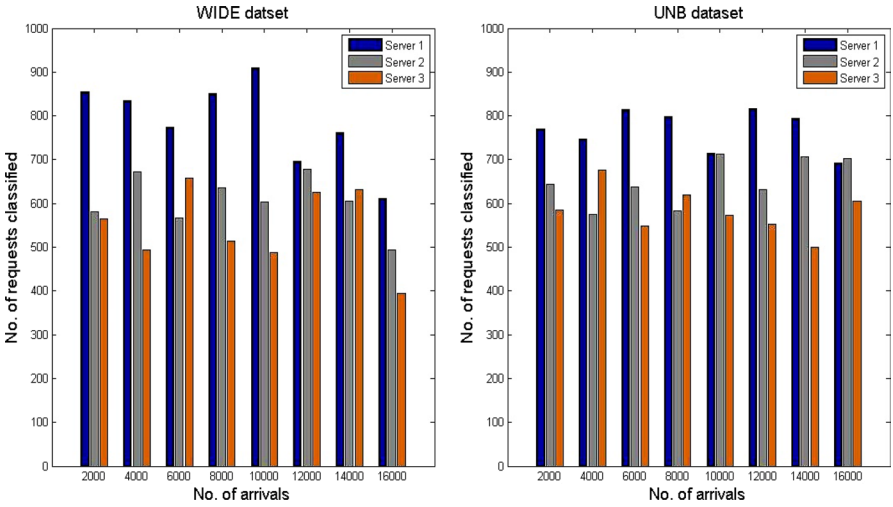

Fig. 8 Number of arrivals versus Number of request classified and forwarded to different servers

The service time of server 1 is less than other two servers in server cluster, as the configuration of server 1 is high, compared to others. So queue size and waiting time are low. So numerous incoming requests are classified as class 1 and forwarded to server 1. Hence in Fig. 8, the number of requests predicted as class 1(to server 1) is greater than other categories. The service time of server 3 is greater than other two servers, so the queue size and waiting time are slightly long in server 3. Hence the number of requests classified as class 3 and forwarded to server 3 are less and it is noticed in Fig. 8.

To evaluate the proposed model in predicting each type of request, the prediction rate of each category is also computed. They are plotted in Figs. 9 and 10. The prediction rate is defined as the ratio of number of requests that are classified correctly in each category to the number of requests predicted in the same category. Inbound traffic characteristics and server status are well defined in the proposed TCSDL model. Moreover the proposed deep architecture discovers the associations between

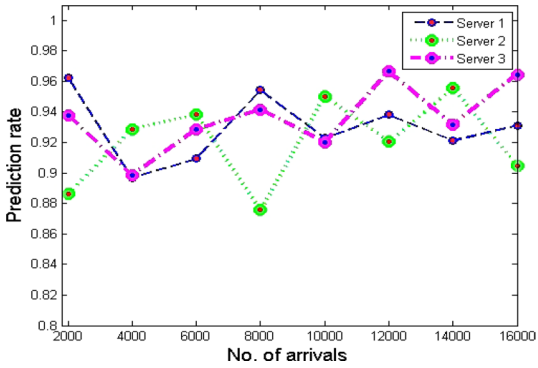

Fig. 9 Number of arrivals versus Prediction rate of each category of request in WIDE dataset

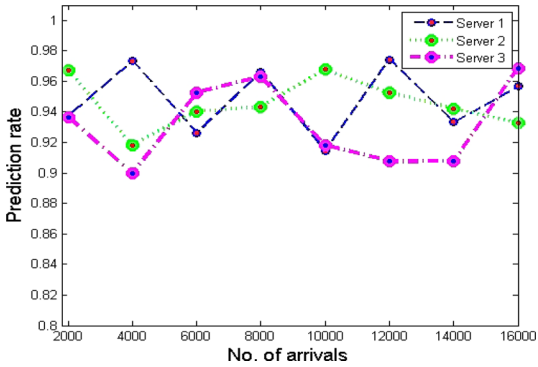

Fig. 10 Number of arrivals versus Prediction rate of each category of request in UNB dataset

traffic flows. Hence the prediction rate of the proposed TCSDL for each category is always above 85% in both datasets in Figs. 9 and 10. Beyond executing the incoming requests, server 2 is assigned with other works. Hence the calculated expected waiting time is not perfect in server 2. This leads to few misclassification in this category. Hence the prediction rate of class 2 (to server 2) is slightly decreased in both datasets. But they are always 85%. Hence it is evident that TCSDL classifies each type of request proficiently.

The objective of the proposed classification model is to improve the QoS of the server cluster and TCSDL classifies the incoming request in such way that they are assigned to the appropriate server and thereby reducing the waiting time of the requests. Hence the performance of TCSDL is assessed in terms of average waiting time. The average waiting time is measured for various arrivals and it is plotted in Fig. 11. The existing SCEW method did not derive the server properties dynamically, whereas in the proposed TCSDL, the server status is considered as a significant feature for classification and they are derived in each session window. Hence the average waiting time of the proposed TCSDL in Fig. 11 is less than the existing static SCEW method. If the service times of the requests are low, then waiting time of upcoming requests are also low. This is observed in WIDE dataset during 6000 arrivals in Fig. 11. In UNB dataset minimum waiting time is observed during 2000 arrival rates.

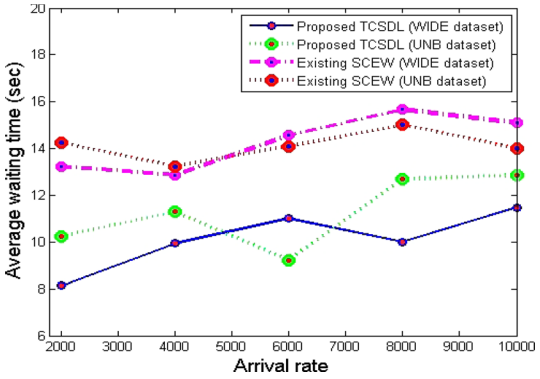

Fig. 11 Arrival rate versus Average waiting time

For various sizes of session windows, the BoFs are constructed and the classification is performed. Each time F-measure and average waiting time are computed and plotted in Fig. 12. F-measure is high during the session window 2500 ms. But here, average waiting time is long. On the other hand, average waiting time is minimum during 500 ms. But here, f-measure is low and the construction of BoF is extremely frequent. Hence the proposed model assigns the duration of session window is 1.5sec, where the average waiting time is not so long; as well as the f-measure is not minimum and it is noticed in Fig. 12.

The performance of classification is improved with extended labelled data and it is observed in Figs. 13 and 14. To reduce the complexity of labelling, the proposed model identifies the correlation between labelled traffic and unlabelled unknown traffic. When it finds similarity, the unlabelled unknown traffic is automatically labelled and added into the extended training dataset. The size of extended training dataset is measured frequently. When substantial increase is observed in size of the extended training dataset, then the proposed TCSDL is trained with extended training dataset. It is observed that the false positive rate is reduced with extended training dataset. The results are plotted in Figs. 13 and 14. From the results, significant increase in F-measure is observed.

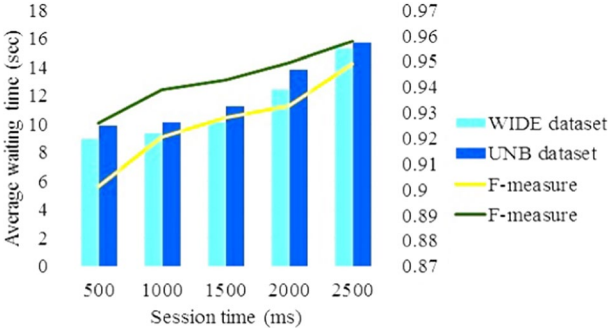

Fig. 12 Session time versus Average waiting time Vs F-measure

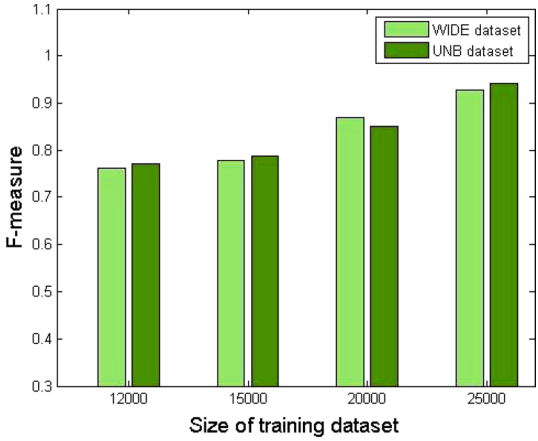

Fig. 13 Size of training dataset versus F-measure

Fig. 14 Learning models versus F-measure

It is inferred from the above results that the proposed traffic classification model using DNN architecture classifies the incoming traffic based the network flow characteristics and status of the server. The proposed classifier recognizes the associations between labelled network traffic and unlabelled unknown traffic in a bag of flow and supports extended labelling.

5 Conclusion

Classification of incoming traffic at server cluster facilitates the improvement of QoS in server cluster. The proposed deep learning classification model categorizes the incoming requests based on the characteristics of inbound network traffic and status of the server cluster. The proposed categorization assists the dispatcher to direct the traffic toward least busy or most appropriate server in the server cluster, this in turn reduces makespan. The proposed extended learning approach effortlessly

defines classes for unlabelled unknown traffic using bag of flow. The incoming traffics are captured using Wireshark. Colaboratory with Keras and Tensorflow is used to develop the proposed deep learning model. It is found from the experimental results that the proposed classifier significantly outperforms the exiting static and machine learning models in terms of classification accuracy and f-measure.

Acknowledgements The authors acknowledge the valuable discussions and suggestions given by Dr. N. P. Gopalan, Professor, National Institute of Technology, Tiruchirappalli for this paper.

References

1. Skala K, Davidovic D, Afgan E, Sovic I, Sojat Z (2015) Scalable distributed computing hierarchy: cloud, fog and dew computing. Open J Cloud Comput 2(1):16–24
2. Taleb T, Samdanis K, Mada B, Flinck H, Dutta S, Sabella D (2017) On multi-access edge computing: a survey of the emerging 5g network edge cloud architecture and orchestration. IEEE Commun Surv Tutor 19(3):1657–1681
3. Shahzadi S, Iqbal M, Dagiuklas T, Qayyum ZU (2017) Multi-access edge computing: open issues, challenges and future perspectives. J Cloud Comput 6(1):30
4. Dipti T, Bhawna M (2016) Svm and naive bayes network traffic classification using correlation information. Int J Comput Appl 147(3):1–5
5. Finsterbusch M, Richter C, Rocha E, Muller JA, Hanssgen K (2013) A survey of payload-based traffic classification approaches. IEEE Commun Surv Tutor 16(2):1135–1156
6. Huang NF, Jai GY, Chao HC, Tzang YJ, Chang HY (2013) Application traffic classification at the early stage by characterizing application rounds. Inf Sci 232:130–142
7. Yuan R, Li Z, Guan X, Li X (2010) An svm-based machine learning method for accurate internet traffic classification. Inf Syst Front 12(2):149–156
8. Hao S, Hu J, Liu S, Song T, Guo J, Liu S (2015) Network traffic classification based on improved dag-svm. In: 2015 International Conference on Communications, Management and Telecommunications (ComManTel). IEEE, pp 256–261
9. Zhang J, Chen X, Xiang Y, Zhou W, Jie W (2014) Robust network traffic classification. IEEE/ACM Trans Netw 23(4):1257–1270
10. Finamore A, Mellia M, Meo M, Rossi D (2010) Kiss: stochastic packet inspection classifier for udp traffic. IEEE/ACM Trans Netw 18(5):1505–1515
11. Zhang Q, Ma Y, Wang J, Li X (2014) Udp traffic classification using most distinguished port. In: The 16th Asia-Pacific Network Operations and Management Symposium. IEEE, pp 1–4
12. Shafiq M, Yu X, Laghari AA, Yao L, Karn NK, Abdessamia F (2016) Network traffic classification techniques and comparative analysis using machine learning algorithms. In: 2016 2nd IEEE International Conference on Computer and Communications (ICCC). IEEE, pp 2451–2455
13. Zhao Y, Wei Z, Zou H (2012) Svm based p2p traffic identification method with multiple properties. Int J Eng Manuf 2(4):1
14. Peng L, Yang B, Chen Y (2015) Effective packet number for early stage internet traffic identification. Neurocomputing 156:252–267
15. Hubballi N, Swarnkar M (2018) *Bitcoding*: network traffic classification through encoded bit level signatures. IEEE/ACM Trans Netw 26(5):2334–2346
16. Xiao X, Li R, Zheng HT, Ye R, KumarSangaiah A, Xia S (2019) Novel dynamic multiple classification system for network traffic. Inf Sci 479:526–541
17. Binfeng W, Jun Z, Zili Z, Lei P, Yang X, Dawen X (2017) Noise-resistant statistical traffic classification. IEEE Trans Big Data. https://doi.org/10.1109/TBDATA.2017.2735996
18. Tongaonkar A, Torres R, Iliofotou M, Keralapura R, Nucci A (2015) Towards self adaptive network traffic classification. Comput Commun 56:35–46
19. Zhang J, Chen C, Xiang Y, Zhou W, Vasilakos AV (2013) An effective network traffic classification method with unknown flow detection. IEEE Trans Netw Serv Manag 10(2):133–147

20. Punitha V, Mala C (2017) Traffic classification for the dispatcher in a server farm based on svm. In: Proceedings of the 2017 International Conference on Intelligent Systems, Metaheuristics and swarm intelligence, pp 93–97
21. Wang Y, Tari Z, HoseinyFarahabady MR, Zomaya AY (2017) Qos-aware resource allocation for stream processing engines using priority channels. In: 2017 IEEE 16th International Symposium on Network Computing and Applications (NCA). IEEE, pp 1–9
22. Lyu Q, Lu X (2019) Effective media traffic classification using deep learning. In: Proceedings of the 2019 3rd International Conference on Compute and Data Analysis, pp 139–146
23. Xu J, Wang J, Qi Q, Sun H, He B (2018) Deep neural networks for application awareness in sdn-based network. In: 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP). IEEE, pp 1–6
24. Lopez-Martin M, Carro B, Sanchez-Esguevillas A, Lloret J (2017) Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access 5:18042–18050
25. Liu B, Xie Q, Modiano E (2020) Rl-qn: a reinforcement learning framework for optimal control of queueing systems. arXiv preprint arXiv:2011.07401
26. Masuda S, He F, Kawabata A, Oki E (2020) Distributed server allocation model with preventive start-time optimization against single failure. In: 2020 IEEE 21st International Conference on high performance switching and routing (HPSR). IEEE, pp 1–6
27. Nguyen TT, Jörg R (2020) Improved bi-criteria approximation schemes for load balancing on unrelated machines with cost constraints. Theor Comput Sci. https://doi.org/10.1016/j.tcs.2020.12.022
28. Cayci S, Gupta S, Eryilmaz A (2020) Group-fair online allocation in continuous time. arXiv preprint arXiv:2006.06852, pp 1–21
29. Tyagi M, Manoria M, Mishra B (2020) Efficient user authentication, server allocation and secure data storage in cloud. Int J Internet Technol Secur Trans 10(1–2):211–228
30. Kaur M, Aron R (2020) Energy-aware load balancing in fog cloud computing. Mater Today Proc
31. Siyun Y, Nelson L, Vidayadhar KG, Haipeng S (2020) Data driven server allocation at virtual computing labs. Queueing Models Serv Manag 3(2):137–166
32. Li D, Asikaburu C, Dong B, Zhou H, Azizi S (2020) Towards optimal system deployment for edge computing: a preliminary study. In: 2020 29th International Conference on Computer Communications and Networks (ICCCN). IEEE, pp 1–6
33. Mukhopadhyay A, Ruffini M (2020) Learning automata for multi-access edge computing server allocation with minimal service migration. In: ICC 2020-2020 IEEE International Conference on Communications (ICC). IEEE, pp 1–6
34. Jaya I, Cai W, Li Y (2020) Rendering server allocation for mmorpg players in cloud gaming. In: 49th International Conference on Parallel Processing-ICPP, pp 1–11
35. Jayasinghe M, Tari Z, Zeephongsekul P, Zomaya AY (2011) Task assignment in multiple server farms using preemptive migration and flow control. J Parallel Distrib Comput 71(12):1608–1621
36. Sreeram I, Vuppala VPK (2019) Http flood attack detection in application layer using machine learning metrics and bio inspired bat algorithm. Appl comput inf 15(1):59–66
37. Prasad KM, Reddy ARM, Rao KV (2017) Bifad: bio-inspired anomaly based http-flood attack detection. Wirel Pers Commun 97(1):281–308
38. Xiao C, Ye J, Esteves RM, Rong C (2016) Using spearman's correlation coefficients for exploratory data analysis on big dataset. Concurr Comput Pract Exp 28(14):3866–3878
39. Chappell L, Combs G (2010) Wireshark network analysis: the official Wireshark certified network analyst study guide. Protocol Analysis Institute, Chappell University
40. Fontugne R, Borgnat P, Abry P, Fukuda K (2010) Mawilab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In: Proceedings of the 6th International COnference, pp 1–12
41. Lashkari AH, Draper-Gil G, Mamun MSI, Ghorbani AA (2017) Characterization of tor traffic using time based features. ICISSP, pp 253–262
42. Pacheco F, Exposito E, Gineste M, Baudoin C, Aguilar J (2018) Towards the deployment of machine learning solutions in network traffic classification: a systematic survey. IEEE Commun Surv Tutor 21(2):1988–2014
43. Boutaba R, Salahuddin MA, Limam N, Ayoubi S, Shahriar N, Estrada-Solano F, Caicedo OM (2018) A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. J Internet Serv Appl 9(1):16
44. Rashmiranjan N, Chandra PU, Kumar DS (2020) A comprehensive review on deep learning-based methods for video anomaly detection. Image Vis Comput 106:104078

45. Li P, Chen Z, Yang LT, Gao J, Zhang Q, Jamal DM (2018) An improved stacked auto-encoder for network traffic flow classification. IEEE Netw 32(6):22–27
46. Blanco V, Japón A, Puerto J (2020) Optimal arrangements of hyperplanes for svm-based multiclass classification. Adv Data Anal Classif 14(1):175–199
47. Punitha V, Mala C (2020) A deep learning approach for detection of application layer attacks in internet. In: Handling Priority Inversion in Time-Constrained Distributed Databases, Chap 10. IGI Global, pp 175–188. https://doi.org/10.4018/978-1-7998-2491-6.ch010
48. Arumugam P, Jose P (2018) Efficient decision tree based data selection and support vector machine classification. Mater Today Proc 5(1):1679–1685