Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

Journal homepage: http://www.science-gate.com/IJAAS.html

Nominate of significant features for unknown internet traffic applications filtering based on a neural network algorithm

Abdulbasit Abbas Mohamed [1,*], Ahmed Hamza Osman [2], Abdelwahed Motwakel [1], Hani Moetque Aljahdali [2]

[1]*Faculty of Computer Science and Information Technology, Omdurman Islamic University, Khartoum, Sudan*
[2]*Faculty of Computing and Information Technology in Rabigh, King Abdulaziz University, Rabigh, Saudi Arabia*

ARTICLE INFO

Article history:
Received 13 July 2020
Received in revised form
17 October 2020
Accepted 22 October 2020

Keywords:
Detection
Classification
Feature selection
Semantic role
Unknown application

ABSTRACT

The evolution of the internet into a large, complex service-based network has posed tremendous challenges for network monitoring and control in terms of how to collect massive volumes of data, in addition to the accurate classification of new emerging applications, such as peer-to-peer networks, streaming content and online games. In this work, machine learning algorithms are used for the classification of traffic into their corresponding applications. Furthermore, this research uses our customized training data set collected from the three institutions' campuses. The effect on the size of the training data set has been considered before examining the accuracy of various classification algorithms and selecting the best from a large amount of data traffic in the network, which has led to delays in performance; therefore, to solve this problem we suggested a distinct approach using multiple neural networks with the feature selection in order to predict and identify known and unknown applications. By applying the proposed method, we get excellent accuracy in the classification of data traffic in the network of up to 99.11%, which leads to improved data traffic in the network and avoids delays.

1. Introduction

Internet traffic describes the sum of data or knowledge available on a site or in another language, which we may claim is a flow of data across the web.

The classification of internet traffic has the ability to handle different types of network problems (Namdev et al., 2015), such as the management of internet traffic applications by identifying an application's basic functionality that has been given to polity. The various solutions include advanced network monitoring, management of network resources, and detection of anomalies, device-specific strategies, and network audits. Therefore, the awareness of the internet at the application level is extremely useful to those who design internet traffic and research long-term internet changes and conditions. In principle, more than 20 years ago, internet measurements found that 70-75% of traffic was online. For various network operations, accurate identifications and predictions of internet-traffic are important, such as for network management and security control, traffic analysis and network preparation, performance, and accounting services delivery (Gunnar et al., 2005). Now, sharing files and applications across the web is often the major influence on data traffic in the network. In order to determine the known and unknown requests in the network (Nguyen and Armitage, 2008) in a given traffic dataset, machine learning can automatically search for and identify useful structural patterns. We investigate the use of multiple neural network algorithms to classify internet traffic.

The advantage of the proposed method is that the model can predict incoming applications and classify them into known and unknown applications in order to reduce web traffic with more accuracy than previous research in the literature.

This article is structured into five parts: Part One is about the motivation and includes an introduction; Part Two addresses the literature linked to the research and process enhancement; Part Three will explain the fundamental structure. The trial architecture and research results, and comprehensive descriptions are to be found in Part Four. The work's conclusions are listed in Part Five.

* Corresponding Author.
Email Address: abdulbasitabbas@gmail.com (A. A. Mohamed)
https://doi.org/10.21833/ijaas.2021.02.015
Corresponding author's ORCID profile:
https://orcid.org/0000-0003-1486-7468

2. Related works

Several methods of traffic clustering have been suggested and tested using stream statistical functionality (Zhang et al., 2013a). McGregor et al. (2004) have suggested aggregating traffic flows into a small number of clusters through the maximization of standards. This is in addition to the statistical characteristics of the algorithm and several full-flow bases. It was found that the algorithm separates traffic into a minimal number of clusters based on the type of traffic rather than the application. Zander et al. (2005) used Auto Class to band traffic flows and suggested a metric for cluster evaluation called intra-class homogeneity. The training method was conducted on a random sampling sub-set of traffic data. The effects of the clustering are tested in terms of accuracy. Bernaille et al. (2006) used a payload analysis tool where the K-means algorithm was applied to traffic clusters and clusters that were marked for applications. The researchers used the first few packets of transmission control protocol (TCP) and the complete stream to describe traffic flows as numerical functions. Erman et al. (2006) evaluated the traffic clustering algorithms, K-means, density-based spatial clustering of applications with noise (DBSCAN), and Auto Class on two empirical traces of data. The authors concluded that the K-means algorithm was more suited to traffic clustering because of its strong overall accuracy and short model construct time. Previous research has shown that where the number of clusters is considerably greater than the number of specifications, traffic clusters may produce high-purity clusters. However, it also contributes to a key problem in the distance between clusters and applications. The payload-based manual mapping was partly bridging the gap or using the method to evaluate payload (Bernaille et al., 2006). Through an automated mapping system, Erman et al. (2006) has also been suggested. Throughout their system, a variety of flows are pre-labeled manually depending on payload. Next, pre-labeled flows together with unlabelled flows are fed into the clustering algorithm K-means. Then, a large number of traffic clusters are mapped to several established applications using the accessible labeled flows. Finally, the closest cluster will be assigned a new traffic flow. Their experimental results showed that the mapping system would achieve high precision with a reasonable array of named flows. These mapping methods can combine traffic clusters of existing flow-based applications, but they cannot fuse the traffic clusters of unknown applications. In recent years, several methods of clustering utilizing payload have also been introduced. Ma et al. (2006) developed three models to capture numerical and systemic dimension data flow sets. The researchers introduced a clustered solution to flow sets and checked their traffic database approach with more than 10 implementations. Zhang et al. (2013b) have combined a statistical signature of Contact and the K-means algorithm to identify unclassified traffic groups dependent on material for use. Wang et al. (2013) suggested using the clustering method for the automatic creation of software signatures dependent on the classifier. They evaluated several supervised classifiers of the clustered traffic generated by the X-means on the payload content of 32 bytes. Such experiments highlight the utility of flow charges to identify specific traffic groups; however, it remains uncertain how to describe the substance and calculate the similarity of traffic clusters, and supervised learning relates to several other works to the classification of traffic on a payload basis. While the classification of traffic is a more reliable way of searching for program signatures in the payload material, it takes a lot of time to derive the signatures manually. To address this issue, Moore and Zuev (2005) have used Classifier Naïve Bayes of kernel estimate and Fast Correlation-Based Filter (FCBF), which has been proposed to be categorized. They used a wide range of 248 functions, including packet series information and the TCP protocol. Moore used nine strategies to identify hand-set info, including gate number, payload header, single packet signature and protocol attribute, first K-Byte payload, host background, etc. Application signatures are obtained in these studies through an in-depth packet-level trace analysis or device procedure documentation (where accessible). The latest developments have seen an attempt to free citizens from the stressful pre-processing phase of labor. Wang et al. (2010) suggested using supervised machine programming to recognize signatures to a variety of technologies automatically. Finamore et al. (2011) suggested code signatures to carry out traffic analysis using g numerical characterization of payload and implemented controlled algorithms, such as the support vector machine (SVM). Existing classification methods based on payload, however, cannot deal with "unknown applications." Zhao et al. (2008) suggested traffic classification real-time feature collection. The different types of characteristics used in the traffic classification are discussed, and the accuracy of various algorithms for traffic selection, in particular the Peer to Peer (P2P) classification of traffic, is evaluated and compared in the classification of traffic. They suggested a real-time function subset to complete the online traffic classification. Cao et al. (2015) suggested that the SVM's classification performance after scaling is better, but the high feature dimension causes the SVM classifier to have a longer training time and higher computational complexity. By this method, we obtain the accuracy of each flow according to the characteristic numbers, and the accuracy would be the maximum in any characteristic number of each traffic flow. Get this number of features, and compose the best subset of features. After feature selection, the average accuracy of all flows reaches 98.69%. Lotfollahi et al. (2020) proposed a deep learning approach that combines both the extraction and classification phases of features into a single system. The proposed scheme, known as the "Deep Packet," can handle both traffic classification in

which network traffic is divided into major groups (e.g., File Transfer Protocol (FTP) and P2P) and application recognition in which end-user applications (e.g., BitTorrent and Skype) are defined. Unlike most current approaches, Deep Packet can recognize encrypted traffic and even differentiate between network traffic, and the Deep Packet architecture employs two deep neural network architectures, architectures for network traffic classification.

3. Proposed model

The essential aim of this paper is to classify and predict network traffic data. The proposed method has four primary steps, which are:

A. Pre-processing, such as data cleansing, outliner, and missing values removal.
B. Dividing the dataset into learning and testing data.
C. Applying the multiple neural network algorithm and feature selection method across the network traffic dataset.
D. Classifying and determining the accuracy of the known applications and unknown applications that affect the network.

The steps of the proposed model are demonstrated in Fig. 1.

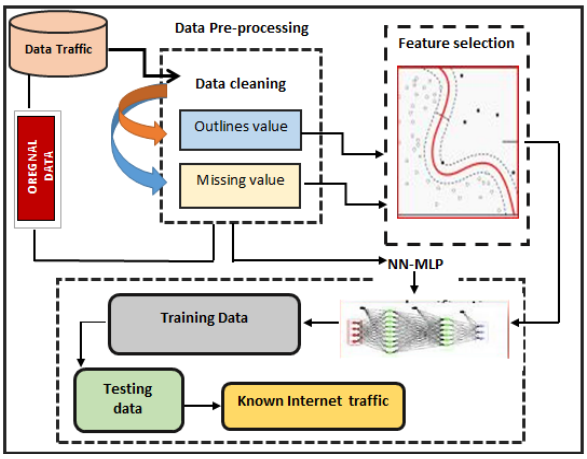

Fig. 1: Proposed model

3.1. Data pre-processing

Data pre-treatment is one of the most important steps in the preparation of the data set before the mining process. In our research, the statistical program software platform, which offers advanced statistical analysis (IBM SPSS), was used to analyze the data, and the hybridmethod consisting ofneural networks and a feature selection method. We used the ten ready datasets collected by a high-performance network screen (Auld et al., 2007). The data were classified according to the entry for each category. The parameters input of the neural network and feature selection that have been used are Interactive, Database, Games, Sevres's, Mail, Www, p2p, Attack, and Media. We used the same 10 datasets that were extracted by Moore and Zuev (2005). 249 different discriminators have been used in our research to define traffic flows, including statistics on flow length, TCP port data, statistics on payload size, and four-part packet transformation. They constructed a flow collection by tracking the day, breaking it into ten blocks of roughly 1,680 seconds (28 min) each. They offered a wider variety of mixtures during the day using the random selection of samples. The dataset comprises specific flow levels in each data-block. Traffic single block of 28-minutes has been captured owing to the higher traffic level. We divided all groups into percentages to determine the accuracy of each percentage of the training and test experiments of the data with 50%, 60%, and 70% for training, 50%, 40%, and 30% for the test, respectively. Accuracy results were extracted from each group, and the average results for these groups have been calculated.

In this phase, the text pre-processing stage contained three sub-stages, which were text chunk, stop words withdrawal, and term stemming. A text chunk partitioned a text archive into sub-sentences. Several of the studies which concentrate on text preparing strategies in various fields incorporate intrusion detection (Sharma et al., 2007). The step of stop terms removal for erasing meaningless terms was utilized. A stemming procedure to delete the attached (suffixes and prefixes) in a term to create its root term was additionally connected. This progression separated the critical terms from the text and disregarded the rest of the terms. This may have influenced the comparability between texts unfavorably.

3.2. Combination stage

In this stage, we combined two techniques: the neural network approach and features selection methodology, to learn about the classifiers.

3.3. Artificial neural networks

Neural networks are the typical depiction of the brain focused on nature neurons that are associated with other neurons to create a network, like "move the hand to pick up the cup." An artificial neural network is normally placed on tables, such that tables n-1 and n+1 will only bind to neurons (Arnx, 2018). We can characterize an artificial neural network like Fig. 2.

Usually, neural networks tend to be converted from left to right. The first layer here is the one that accesses outputs. There are two internal layers that do some algebra (known as invisible layers) and one final layer that includes all possible inputs. Do not mess around with the "+1"s at the bottom of each line. It's labeled "bias."

Every neuron's operations are quite simple (Fig. 3) (Arnx, 2018).

Firstly, it applies the meaning of an earlier section that is correlated with each neuron. There are three

neuron outputs in Fig. 2 (x1, x2, and x3), so our neuron is related to the three neurons of the previous section. By applying this value, this quantity is compounded by another variable named "weight" (w1, w2, w3). That decides how they interact amid the two neurons. Increasing neuronal interaction has its own weight, and these are the only principles that can shift in the course of learning.

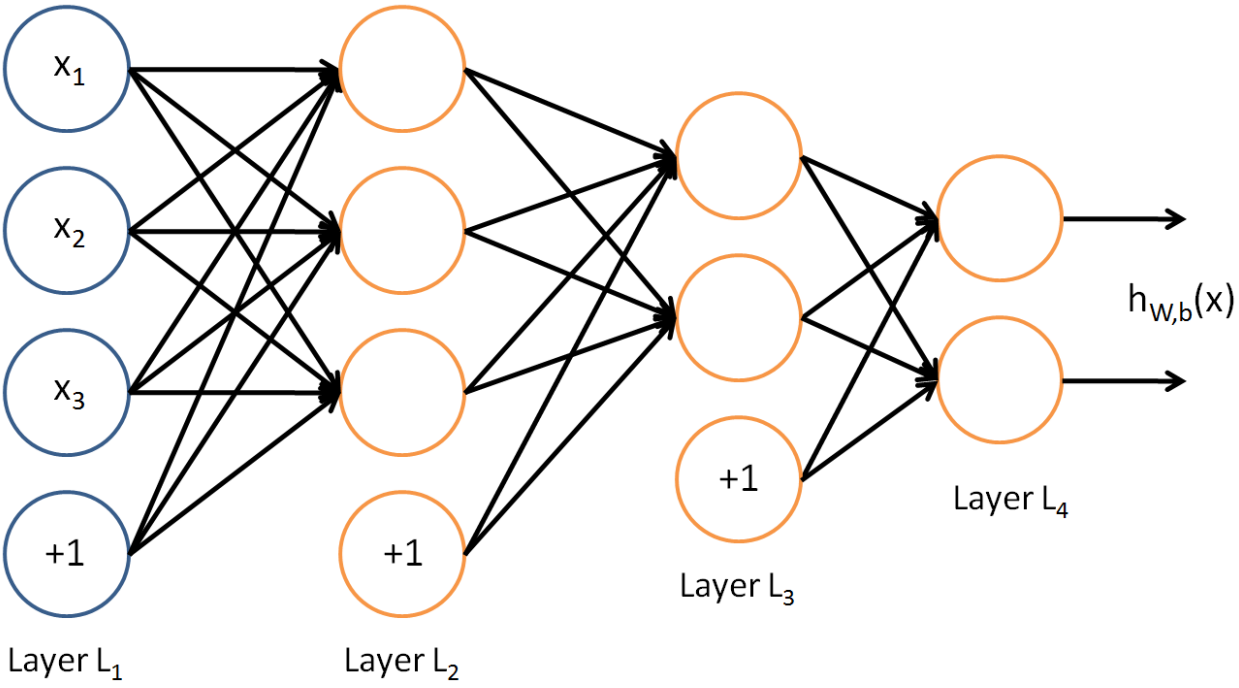

Fig. 2: Representation of multiple neural networks

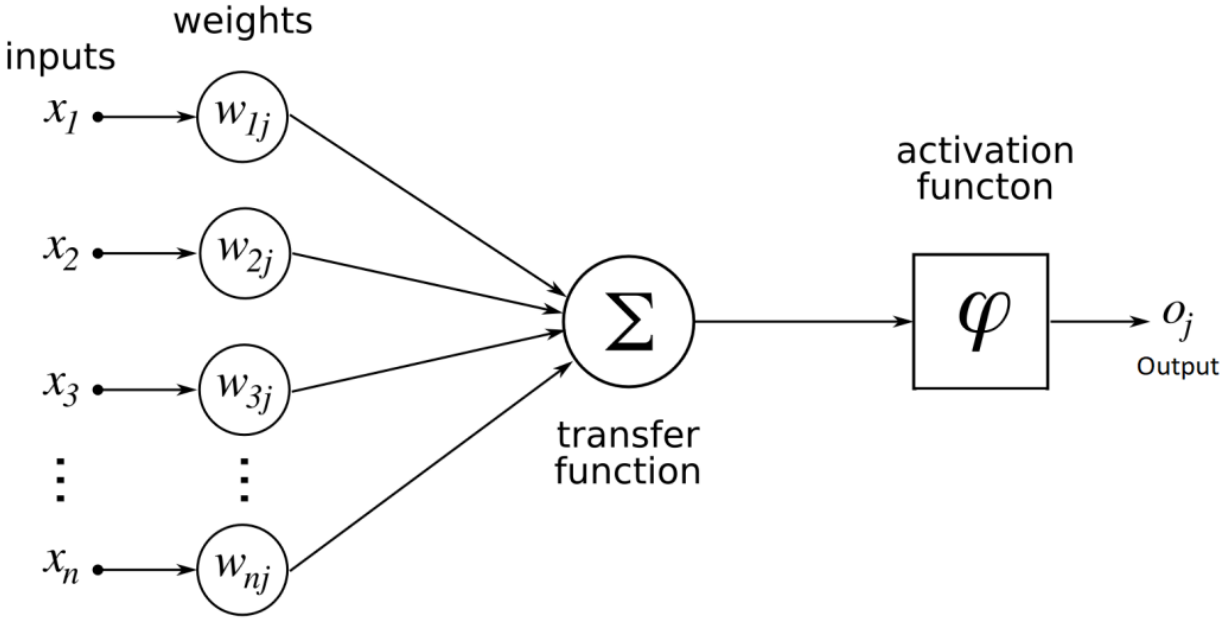

Fig. 3: The operations performed by multiple neural networks

In contrast, the estimated total cost can be added to a discrimination variable. It is not an output that comes from a single neuron that is selected before the learning process, so it can be helpful to the network.

This is all done by a neuron. One has to take all the values compounded by their respective weight from attached neurons, bind them, and add an activation mechanism to it. The neuron will then give the new value to other neurons.

The parameters for the Neural Network algorithm have been selected based on the nature of the data on the network. The input parameters have been tuned as features of the applications on the network, while the output has been adjusted as a target of classification features to known and unknown applications based on the weighted parameters in the hidden layers.

The neural network moves to the next row after each neuron of a column has been made. The last obtained value must eventually be one that can be

used to evaluate the output wanted. This is how the learning cycle functions: First, note that it returns an output when an input is provided to the neural network. It cannot get the right output on its own at the first attempt (except with luck), and this is why each input comes with its tag during the learning phase, indicating what should be the performance of the neural network. If the option is the right one, the variables will be preserved, and the corresponding data will be given. If the output received does not match the tag, however, weights will be modified. These are the only factors in the learning phase that can be modified. This mechanism can be interpreted as several keys, which are converted into different possibilities each time an input is not correctly calculated.

A complex process called "backpropagation" is performed to decide what weight is best to change. We are not going too far longer on this, as the neural network we are trying to create does not use the same method, so it is about going back to the neural network and testing each relationship to see if the output would respond as a consequence of a weight shift.

Eventually, there is a final variable to know how to monitor neural network learning: the "learning level." It defines how quickly the neural network is going to know or, more precisely, how the weight will shift, slowly or in bigger steps. Ultimately, this variable is a good value.

Now that we understand the fundamentals, we can test the neural network we are going to create. This design allows two classes to be separated by an easy category. Let's see a quick example (which has little value except to understand) to better understand the possibilities and limitations.

When we substitute the "trues by 1 and the falsies by 0" and put the four options on a graph as coordinate points, it is clear that the two "false" and "true" final classes can be separated by a single line. This can be done by a perceptron. A neural network can be built from scratch with Python (3.x in the following example):

```
Import numpy, random, os
lr=1 #learning rate
bias=1 #value of bias
weights=[random.random (), random.random (), random.random ()] #weights generated in a list (3 weights in total for 2 neurons and the bias)
```
(1)

Put simply, libraries and parameter values can be defined at the start of the program, and a list containing the values of the weights to be changed can be created at random.

Below is a structure that determines the output neuron function. It needs three variables (the two values of the neuron and the output predicted). "OutputP" is the variable that corresponds to the perceptron's output. Then, we compute the error, which is used straight afterward to change the weights of each connection to the output neuron (Arnx, 2018).

```
for i in range (50):
Perceptron (1,1,1) #True or true
Perceptron (1,0,1) #True or false
Perceptron (0,1,1) #False or true
Perceptron (0,0,0) #False or false
```
(2)

We are building a circle loop repeating every situation several times by the neural network. This part is the process of reading. The number of iterations is selected based on the reliability we need. We have to be mindful, however, that too many iterations could result in the network being over-fitted, allowing it to concentrate too much on the instances being handled, so it cannot get the right performance of a case that it did not see during its training phase.

Nonetheless, our situation here is very different because there are only four options, and we send all of them during their learning phase to the neural network. A perceptron should give the right output without ever seeing the case that is being treated (Arnx, 2018).

```
X=int (input())
Y= int (input())
Outputp=x*weights[0]+y*weights[1]+bias*weights[2]
If outputp>0: #activation function
 Outputp=1
Else:
Outputp=0
```
(3)

Lastly, we will ask the consumer to enter the values to test if the perceptron is operating. This is the study phase.

In this case, it is useful to use the activation function, Heaviside. All values are taken back to exactly 0 or 1, as we are finding a fake or a real value. We may try to get a decimal number between 0 and 1 with a sigmoid feature, typically very close to one of those limits.

```
Outputp=1/1+numpy.exp(-output))#sigmoid function
```
(4)

We could also save the weights already determined in a file by the neural network to use later without any additional stage in the learning experience. This is done for a broader project and in that cycle will last days or weeks. The study suggested a multiple neural network technique to predict and filter data traffic on the network to identify unknown applications through the physical network. Multiple neural network algorithms were used to perform a scientific experiment to assess the accuracy of internet traffic for potential enhancement. It has been demonstrated that several neural network model applications can be used to predict and process high accuracy network data traffic, as we will be doing later.

3.4. Feature selection algorithms

In this section, we present the classical selection algorithm: a forward selection of features (Mao, 2002). Then, we examine selfish forward algorithm

variants to boost computational efficiency without the risk of losing so much accuracy.

The feature selection process begins by analyzing all sub-sets of features consisting of one attribute for data. In other words, we start by measuring the sub-sets of one element's Leave-One-Out Cross-Validation (LOOCV) error, [X1], [X2], ..., [XM], where M is the input dimension, so we can find the best individual component, X(1). The complete selection process for selecting the function up to m attributes:

1. Collect a specific domain data set for the training.
2. Shuffle this set of data.
3. Divide it into P partitions (say P=20)
4. For each partition (i=0, 1... P-1)
a. Let OuterTrainset (i)=all partitions excepti.
b. Let OuterTestset (i)=the i'th partition
c. Let InnerTrain (i)=randomly chosen 70% of the OuterTrain-set (i)
d. Let InnerTest (i)=the remaining 30% of the OuterTrainset(i).
e. For j=0, 1, ..., m
Search for the best feature set with j components, fsij. using Leave-one-out on Inner Train (i)
Let InnerTestScoreij=RMS score of fsij on Inner Test (i). End loop of (j).
f. Select the fsij with the best inner test score.
g. Let OuterScorei=RMS score of the selected feature set on OuterTestset (i)
End of loop of (i).
Return the mean Outer Score. (5)

First, forward selection would find two strong subset components, X (1), and another function of the rest attributes of M-1 data. Therefore, there are M-1 pairs in total. Suppose X (2) is the other attribute besides X (1) in a strong set. Then, the input subsets are tested with three, four, and more functions. The safest m-function subset is the m-tuple composed of X (1), X (2), ..., X (m), according to the forward selection, while the overall best collection of features is the winner of all measures of the M. If the cost of a LOOCV evaluation of I features is C (i), then the computational expense of choosing a sub-set of size m out of the total M input attributes would be:

$$MC(1)+(M-1)C(2)+...+(M-m+1)C(m) \quad (6)$$

Liu and Motoda (2007) estimated the cost of predicting one-nearest-neighbor as function, using a kd-tree with j inputs, is O (j log N) where N is the number of data points. Therefore, the expense of measuring the mean leave-one-out mistake, including calculations of N, is O (j N log N). So, the maximum expense of using the aforementioned equation to pick the function is O (m2 M N log N).

We can also use an exhaustive search to find the best overall output feature collection. The exhaustive search starts by searching for the best one-component subset of input features, which is similar to the forward selection algorithm. Instead, the strongest two-component subset of features that may consist of any pair of input features will be identified. It then moves to find the best triple out of all the combinations of each production of three functions, etc. The comprehensive quest meaning is as follows (Arnx, 2018):

$$MC(1)+ (M/2)C(2)+...+(M/m)C(M) \quad (7)$$

The collection advances are far cheaper than the comprehensive quest.

Nevertheless, the forward option will suffer because of its greed. For example, if X (1) is the best individual function, then there is no assurance that either [X (1), X (2)] or [X (1), X (3)] would have to be better than [X (2), X (3)]. Thus, a forward selection algorithm may pick a feature set other than the one selected by exhaustive quest. Estimating a query with a poor set of features of the input: Xq=[x1, x2, ..., xM] can vary significantly from the true Yq.

4. Experimental design

This experiment was aimed at identifying and filtering unknown internet traffic applications. We used the ready dataset that was gathered via a high-efficiency network panel. We used their minimal loss and capture of complete payload to a disk with a resolution of more than 35 nanoseconds for time-stamps. They examine data in time from one website over several different periods of time. This place is an investigational center hosting approximately 1,000 internet-connected users through a Gigabit Ethernet full-duplex connection. For each traffic collection, full-duplex traffic on this link has been controlled. The location they were looking at houses many biology-related buildings, collectively regarded as a Genome Campus. There are three organizations on-site that hire about 1,000 scholars, managers, and professional personnel. This is a campus connected to the internet with a full-duplex Gigabit Ethernet link. Our screen was put on this internet connection. For each traffic array, traffic was tracked for a complete 24-hour, weekday duration, and for all connections.

Appropriate input data are needed for the analysis of data using the neural network technique. To this end, we capitalized on the trace data identified and categorized. This confidential data was further reduced, with each having about 25,000–65,000 items (flows) separated into ten periods of equivalent time. In addition, each data set was used as a training set and tested against the remaining data sets to determine the efficiency of the neural network methodology, allowing for estimation of the average classification accuracy. In each round, the data were divided into three clusters (Osman and Aljahdali, 2017) (70%, 60%, and 50%) for the learning process and (30%, 40%, and 50%) for research. Each learning and testing takes the following traffic into account.

4.1. Traffic categories

One of the fundamental matters for the classification movement is the selection of categories

from the flowing data to perform the classification on. In this research, we use the most popular categories of users, such as (BULK, DATABASE, INTERACTIVE, MAIL, SERVICES, WWW, P2P, ATTACK, GAMES, MULTIMEDIA), examples of which are given to them. In Table 1, these categories are not all traffic data only. They are popular categories, and therefore we ran experiments on them. Each category has unique characteristics and features (such as the source and destination ports), a certain amount of information, and its own behavior. Together, this information and data form the important values for input to make classifications for data traffic within the network. A probabilistic classification approach for internet traffic is used for specific classes to determine the flow characteristics and shunt of the potential layer; for example, flow is classified with a probability of 0.9 that it is a game, 0.1 bulk, and 0.2 that it is www. Flow is classified with the highest likelihood, and in the example, flow is classified as a game category because it is the highest probability. Table 1 shows the network traffic allocation to each category (Moore and Zuev, 2005).

Our key objective for classification is the flow, and for the research addressed in this extended abstract, we restricted our description of the flow to a maximum TCP flow–that is, all the packets between two hosts–for a specific tuple that we limit to complete flows, those that validly start and finish.

Table 1: Network traffic category

Classification	Example Application
Bulk	FTP, TFTP
Database	Postgres, sqlnet, oracle, Ingres
Interactive	ssh, klogin, rlogin, telnet
Mail	Imap, pop2/3, SMTP
Services	X11, DNS, ident
Www	www
P2P	kaZaA, BitTorrent, GnuTella
Attack	Internet worm and virus Attacks
Games	Microsoft direct play
Multimedia	Windows Media Player, Real

The illustration of the classification of the discriminators for every entity can be shown as:

- Data Flow time
- Port with TCP and unshielded twisted pair (UTP)
- Intra-arrival packet time (mean, variance)
- Element of payload (mean, variance)
- Active Entropic Bandwidth
- Fourier transfer of inter-arrival time for packets

An example of discriminator classification objects has been presented in order to classify the scheme that involves defining each element's characterization. By using these variables, the classifier assigns an entity to a class because of its potential to enable discrimination between classes. Such object-describing parameters are used as discriminators. 249 different discriminators (Moore and Zuev, 2005) have been used in our research to define traffic flows, including Statistics on flow length, TCP port data, statistics on payload size, and four-part packet transformation.

4.1.1. First experiment

The study used the experiment that was described before and after combination using neural network methods to evaluate the improvement of the proposed method. Before the upgrade, the filter tests were obtained using a neural network, which is only 98.54%, 98.55%, and 98.60% of learning data and 98.66%, 98.48%, and 98.58% of research data, respectively, in Table 2. On the other hand, filtering performance outcomes after enhancement using the combination approach between the neural network and the potential rating algorithm was 98.68%, 98.98%, 98.93% for training data and 99.10%, 99.11%, and 99.04% for testing data.

The outcomes of the tests are determined as:

$$\text{Accuracy}=\frac{TP+TN}{TP+TN+FP+FN} \quad (8)$$

where, True Positive (TP): Correctly listed the number of documented applications and the available unknown-applications. False Positive (FP): The number of executables known to the applications classified as unknown. Real Negative (TN): Wrongly listed the number of established applications and unknown-applications executable. False-negative (FN): Number of executables unknown-applications listed as known applications.

Table 2: Results of the neural network without feature selection algorithm

Algorithm	Accuracy		Error	
	Training	Testing	Training	Testing
Neural network(mlp)	98.54 (50% size)	98.66 (50% size)	1.46	1.34
Neural network(mlp)	98.55 (60% size)	98.48 (60% size)	1.45	1.52
Neural network(mlp)	98.60 (70% size)	98.58 (70% size)	1.4	1.42

Figs. 4, 5, and 6 demonstrate that the testing filter accuracies of the multiple layers perceptron (MLP) neural network have 50%, 60%, and 70%, respectively. The results of the neural network only with the latest training and testingtests algorithm are presented. This is the graph obtained for the neural network training and testing performance before using the feature selection technique in Table 2. As for the classification using a neural network, a successful filter for the training data was obtained 98.54% with a sample size of 50% and filter performance of 98.66% for processing data with a data size of 50%. Expressively, the classification performance outcomes obtained by the neural network were 98.54%, 98.55%, and 98.60%, respectively, for learning tests of 50%, 60%, and 70%, and 98.66%, 98.48%, and 98.58%, respectively, for research studies of 50%, 60%, and 70%.

4.1.2. Second experiment

The performance obtained after integrating the neural network with feature selection is described in Table 3. The accuracy of the filtering process is clarified by the values of results. There are various results that have been derived from the neural network. Such findings have been improved by using the features selection method. The neural network with a feature selection approach obtained better accuracy than the neural network alone.

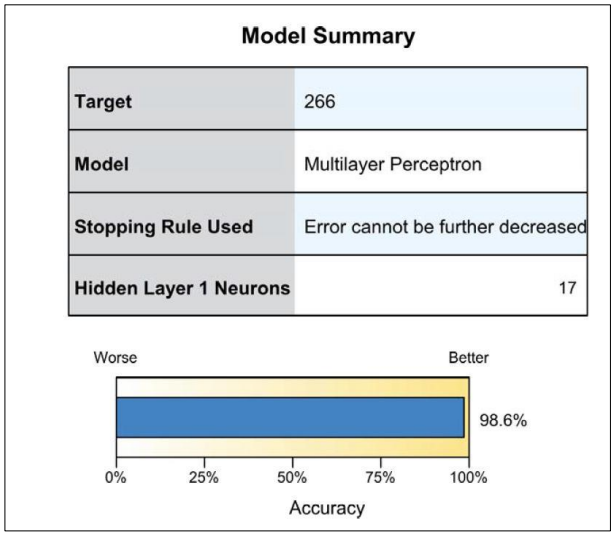

Fig. 4: Filter accuracy testing neural network 50%

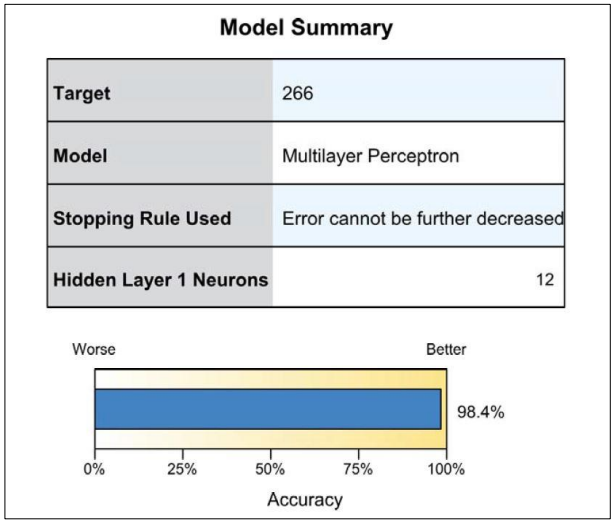

Fig. 5: Filter accuracy testing neural network 60%

Figs. 7, 8, and 9 display the reliability of neural network tests with a feature selection method based on important features with the best category in Table 3 for training and testing experiments. After using the feature selection algorithm with important features, the figures obtained represented test results of the neural network with the first collection of data. The classification using a neural network without a feature selection method for testing data obtained optimum filtering for an average of 10 datasets is 98.66%, 98.48%, and 98.58 with data size of 50%, 60%, and 70%, respectively (Table 4). The classification using a hybrid method (neural network with feature selection) for testing data obtained optimum filtering for an average of 10 datasets is 99.10, 99.11, and 99.04 for research experiments of 50%, 60%, and 70%, respectively (Table 5). From this, we conclude that the use of the hybrid method led to better results than the use of the neural network method alone.

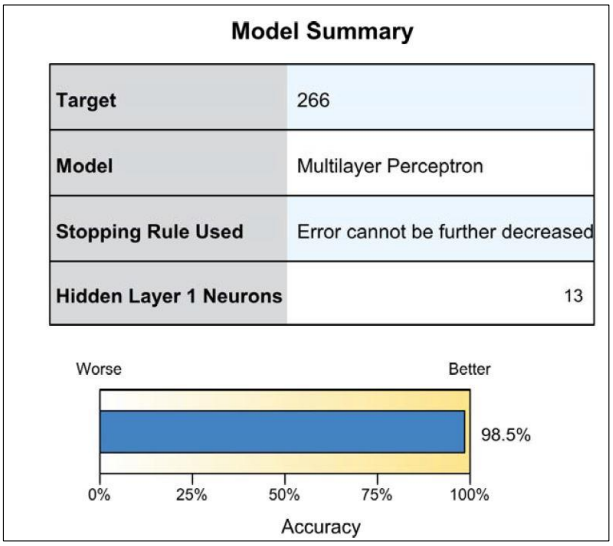

Fig. 6: Filter accuracy testing neural network 70%

Table 3: Results of neural network with feature selection algorithm

Algorithm	Accuracy		Error	
	Training	Testing	Training	Testing
Neural network(mlp) and Feature selection	98.68 (50% size)	99.10 (50%size)	1.32	0.9
Neural network(mlp) and Feature selection	98.98 (60% size)	99.11 (60%size)	1.02	0.89
Neural network(mlp) and Feature selection	98.93 (70%size)	99.04 (70%size)	1.07	0.96

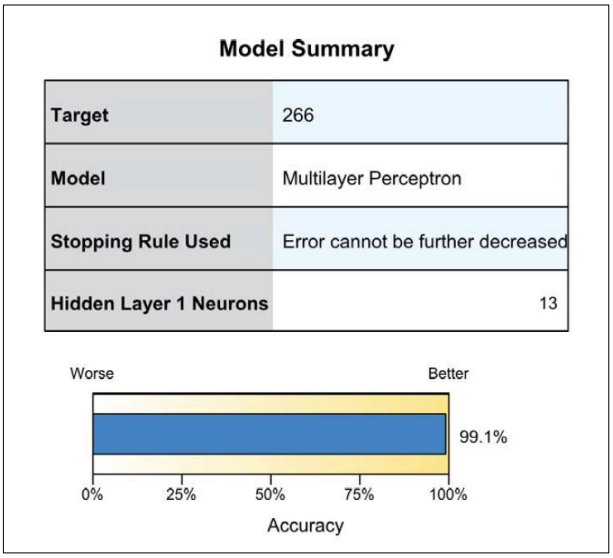

Fig. 7: Filter accuracy testing neural network with feature selection 50%

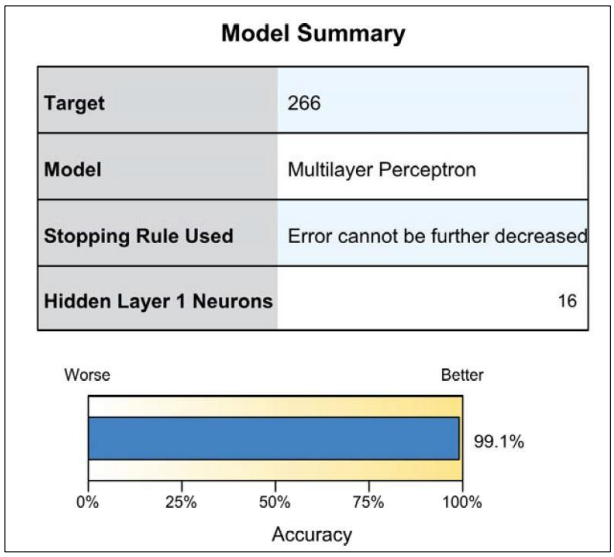

Fig. 8: Filter accuracy testing neural network with feature selection 60%

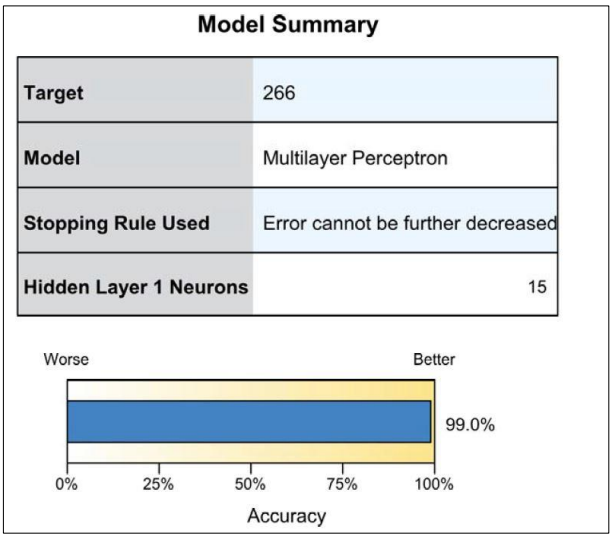

Fig. 9: Filter accuracy testing neural network with feature selection 70%

Table 4: Experimental results without feature selection

Dataset use the neural network (MLP) before feature selection						
Dataset NN-MLP	ACCURACY-TR-50%	ACCURACY-TS-50%	ACCURACY-TR-60%	ACCURACY-TS-40%	ACCURACY-TR-70%	ACCURACY-TS-30%
DATASET-1	98.2	98.6	98.6	98.9	98.4	98.7
DATASET-2	98.7	98.7	98	98.6	98.8	98.8
DATASET-3	98.9	98.9	98.9	98.7	98.9	98
DATASET-4	98.7	98.4	97.7	98.3	98.2	98.6
DATASET-5	98.5	98.9	98.8	98.7	98.9	98.4
DATASET-6	98.3	98.8	98.6	97.4	98.4	98.7
DATASET-7	98.5	98.5	98.9	99	98.8	98.8
DATASET-8	98.9	98.7	98.9	98.1	98.8	98.9
DATASET-9	98	98.5	98.2	98.3	98	98.3
DATASET-10	98.7	98.6	98.9	98.8	98.8	98.6
AVARAGE	98.54	98.66	98.55	98.48	98.60	98.58

Table 5: Experimental results with feature selection

Dataset using neural network (MLP) after feature selection						
Dataset +NN-MLP	ACCURACY-TR-50%	ACCURACY-TS-50%	ACCURACY-TR-60%	ACCURACY-TS-40%	ACCURACY-TR-70%	ACCURACY-TS-30%
DATASET-1	98.3	99	98.9	98.9	98.4	99.1
DATASET-2	99.2	99.3	99	99.1	99.4	99
DATASET-3	98.9	99.1	99.1	99.2	99.1	99
DATASET-4	98	99	99	99.1	98.4	98.7
DATASET-5	99.2	99.1	98.8	98.9	99.2	99.3
DATASET-6	98.1	99.2	99.2	99.2	99	98.9
DATASET-7	98.7	99.1	99.3	99.4	99.3	99.2
DATASET-8	99	99.1	99	99.1	99.1	99.1
DATASET-9	98.4	99.2	98.6	99.3	98.5	99
DATASET-10	99	98.9	98.9	98.9	98.9	99.1
AVARAGE	98.68	99.10	98.98	99.11	98.93	99.04

For comparative approaches, most authors used the methodology of statistical significance for the t-test. The t-tests use the hybrid approach to assess statistical significance. Among the findings obtained from Experiment 1 using the neural network-mlp and Experiment 2 using the hybrid approach include the neural network and feature selection technique revealed improvements. Table 5 illustrates standard deviations, certain events, mean values, standard errors, and sensitive tests for pairs with pre and post factors neural network enhancementwith feature selection method compared to paired samples t-test. The paired-sample test measures the mean of two variables reflecting the same unit at various periods. The two mean value variables are shown in the table of paired samples. Since the paired samples t-test measures the value of the two variables, understanding what the mean values are is important. A small t-test meaning value typically less than 0.05 indicates that the two variables differ. The outcome of the t-test is 0.0183; this condition has been stressed in estimation steps, indicating that the hybrid methodology (neural network with a collection of features) has obtained important results on the accuracy of the study. This discrepancy is deemed statistically relevant by traditional standards. Table 6 shows t-test comparison results between the neural network algorithm before and after feature selection.

Table 6: T-test comparison results between the neural network algorithm before and after feature selection

Method	Differences result in 70%, 60%, and 50% dataset between the neural network and feature selection					t	Sig. Value
	mean	Std. Deviation	Std. Err Mean	95% Confidence Interval of the Difference			
				Lower	upper		
Neural network and feature selection	0.51	0.037	0.02	0.208	0.81	7.285	0.018

The proposed approach implemented the algorithm of feature selection to boost the process of the neural network. In the classification process, only the most appropriate features as selected by the feature selection method were used. The findings of the experimental test dataset showed that better results were obtained by the overall performance of the proposed method. The hypothesis presented the idea that the selection technique can improve the quality of classification. The proposed method's emphasis was changed so that attention was paid to before and after the combination phase to analyze the changes made by the proposed method.

The comparison between the proposed method and state of the art is illustrated in Table 7.

Table 7: Comparison between the proposed method and a state of the art

Method	Accuracy Results	Reference
K-Means Clustering algorithms	92.4%	(Erman et al., 2006)
Bayesian Neural Networks method	95%	(Auld et al., 2007)
support vector machine (SVM)	97.8%	(Kim et al., 2008)
Statistical Bias Correction Kit (SBCK) method	96%	(Wang et al. 2013)
Kernel-Based Extreme method	96.27%	(Ertam and Avcı, 2016)
incremental learning method	96.00%	(Sun et al., 2017)
Transfer learning model (TrAdaBoost)	94.6%.	(Sun et al., 2018)
C5.0 decision tree classifier	97.4%	(Oudah et al., 2019)
Proposed method	99.11%	

We noted that the proposed method achieved better performance results in terms of classification accuracy.

We noted that the shortcoming of the proposed method is that the prediction model can classify only offline applications rather than online applications. The time of classification needs to be improved if the model is to be upgraded to work online.

5. Conclusion

This research attempted to solve the issue of the identification of internet traffic. The study suggested that the hybrid method of multiple neural network technique and the feature selection method can be used for predicting and filtering network data traffic to classify the unknown applications through the physical network. A scientific experiment has been conducted using multiple neural network algorithms to determine the reliability of internet traffic for future enhancement. It has been shown that the applications of the multiple neural network models can be used to predict and filter the network data traffic with high accuracy.

The data was collected and divided into 10 groups, and each data group was divided into percentages to be based on the experiment, as follows: 50%, 60%, and 70% for training, and in return for the same data 50%, 40% and 30% for the testing. As for the percentage of 50% for testing, the results showed a clear improvement in classification and verification with an accuracy rate of 99.10% and with an error rate not exceeding 0.9, and thus we would have improved the accuracy of the classification of unknown internet traffic applications by using multiple neural network algorithms with the feature selection method.

Compliance with ethical standards

Conflict of interest

The author(s) declared no potential conflicts of interest with respect to the research, authorship, and/or publication of this article.

References

Arnx A (2018). Understand and create a Perception. In the First Neural Network for Beginners Explained (with code), CS Student, France.

Auld T, Moore AW, and Gull SF (2007). Bayesian neural networks for internet traffic classification. IEEE Transactions on Neural Networks, 18(1): 223-239. https://doi.org/10.1109/TNN.2006.883010 **PMid:17278474**

Bernaille L, Teixeira R, Akodkenou I, Soule A, and Salamatian K (2006). Traffic classification on the fly. ACM SIGCOMM Computer Communication Review, 36(2): 23-26. https://doi.org/10.1145/1129582.1129589

Cao J, Fang Z, Zhang D, and Qu G (2015). Network traffic classification using feature selection and parameter optimization. Journal of Communications, 10(10): 828-835. https://doi.org/10.12720/jcm.10.10.828-835

Erman J, Arlitt M, and Mahanti A (2006). Traffic classification using clustering algorithms. In the 2006 SIGCOMM Workshop on Mining Network Data, Association for Computing Machinery, Pisa, Italy: 281-286. https://doi.org/10.1145/1162678.1162679

Ertam F and Avcı E (2016). Network traffic classification via kernel based extreme learning machine. International Journal of Intelligent Systems and Applications in Engineering, 4(Special Issue): 109-113. https://doi.org/10.18201/ijisae.267522

Finamore A, Mellia M, and Meo M (2011). Mining unclassified traffic using automatic clustering techniques. In the International Workshop on Traffic Monitoring and Analysis, Springer, Vienna, Austria: 150-163. https://doi.org/10.1007/978-3-642-20305-3_13

Gunnar A, Abrahamsson H, and Söderqvist M (2005). Performance of traffic engineering in operational IP networks–an experimental study. In the International Workshop on IP Operations and Management, Springer, Barcelona, Spain: 202-211. https://doi.org/10.1007/11567486_21

Kim H, Claffy KC, Fomenkov M, Barman D, Faloutsos M, and Lee K (2008). Internet traffic classification demystified: Myths, caveats, and the best practices. In the 2008 ACM CoNEXT Conference, Association for Computing Machinery, Madrid, Spain: 1-12. https://doi.org/10.1145/1544012.1544023

Liu H and Motoda H (2007). Computational methods of feature selection. CRC Press, Boca Raton, USA. https://doi.org/10.1201/9781584888796

Lotfollahi M, Siavoshani MJ, Zade RSH, and Saberian M (2020). Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Computing, 24(3): 1999-2012. https://doi.org/10.1007/s00500-019-04030-2

Ma J, Levchenko K, Kreibich C, Savage S, and Voelker GM (2006). Unexpected means of protocol inference. In the 6th ACM SIGCOMM Conference on Internet Measurement, Association for Computing Machinery, Rio de Janeiro, Brazil: 313-326. https://doi.org/10.1145/1177080.1177123

Mao KZ (2002). Fast orthogonal forward selection algorithm for feature subset selection. IEEE Transactions on Neural Networks, 13(5): 1218-1224. https://doi.org/10.1109/TNN.2002.1031954 **PMid:18244519**

McGregor A, Hall M, Lorier P, and Brunskill J (2004). Flow clustering using machine learning techniques. In the International Workshop on Passive and Active Network Measurement, Springer, Antibes, France: 205-214. https://doi.org/10.1007/978-3-540-24668-8_21

Moore AW and Zuev D (2005). Internet traffic classification using bayesian analysis techniques. In the ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, Association for Computing Machinery, Banff, Canada: 50-60. https://doi.org/10.1145/1064212.1064220

Namdev N, Agrawal S, and Silkari S (2015). Recent advancement in machine learning based internet traffic classification. Procedia Computer Science, 60: 784-791. https://doi.org/10.1016/j.procs.2015.08.238

Nguyen TT and Armitage G (2008). A survey of techniques for internet traffic classification using machine learning. IEEE Communications Surveys and Tutorials, 10(4): 56-76. https://doi.org/10.1109/SURV.2008.080406

Osman AH and Aljahdali HM (2017). Diabetes disease diagnosis method based on feature extraction using K-SVM. International Journal of Advanced Computer Science and Applications, 8: 236-244. https://doi.org/10.14569/IJACSA.2017.080130

Oudah H, Ghita BV, and Bakhshi T (2019). A novel features set for internet traffic classification using burstiness. In the International Conference on Information Systems Security and Privacy, Prague, Czechia: 397-404. https://doi.org/10.5220/0007384203970404

Sharma A, Pujari AK, and Paliwal KK (2007). Intrusion detection using text processing techniques with a kernel based similarity measure. Computers and Security, 26(7-8): 488-495. https://doi.org/10.1016/j.cose.2007.10.003

Sun G, Li S, Chen T, Su Y, and Lang F (2017). Traffic classification based on incremental learning method. In the International Conference on Advanced Hybrid Information Processing, Springer, Harbin, China: 341-348. https://doi.org/10.1007/978-3-319-73317-3_40

Sun G, Liang L, Chen T, Xiao F, and Lang F (2018). Network traffic classification based on transfer learning. Computers and Electrical Engineering, 69: 920-927. https://doi.org/10.1016/j.compeleceng.2018.03.005

Wang Y, Xiang Y, and Yu SZ (2010). An automatic application signature construction system for unknown traffic. Concurrency and Computation: Practice and Experience, 22(13): 1927-1944. https://doi.org/10.1002/cpe.1603

Wang Y, Xiang Y, Zhang J, Zhou W, Wei G, and Yang LT (2013). Internet traffic classification using constrained clustering. IEEE Transactions on Parallel and Distributed Systems, 25(11): 2932-2943. https://doi.org/10.1109/TPDS.2013.307

Zander S, Nguyen T, and Armitage G (2005). Automated traffic classification and application identification using machine learning. In The IEEE Conference on Local Computer Networks 30th Anniversary, IEEE, Sydney, Australia: 250-257. https://doi.org/10.1109/LCN.2005.35 **PMid:15351913**

Zhang J, Chen C, Xiang Y, and Zhou W (2013a). Robust network traffic identification with unknown applications. In the 8th ACM SIGSAC symposium on Information, computer and communications security, Association for Computing Machinery, Hangzhou, China: 405-414. https://doi.org/10.1145/2484313.2484366

Zhang J, Xiang Y, Zhou W, and Wang Y (2013b). Unsupervised traffic classification using flow statistical properties and IP packet payload. Journal of Computer and System Sciences, 79(5): 573-585. https://doi.org/10.1016/j.jcss.2012.11.004

Zhao JJ, Huang XH, Qiong SUN, and Yan MA (2008). Real-time feature selection in traffic classification. The Journal of China Universities of Posts and Telecommunications, 15: 68-72. https://doi.org/10.1016/S1005-8885(08)60158-2