Neural Processing Letters (2021) 53:3497–3523
https://doi.org/10.1007/s11063-021-10534-6

CSCNN: Cost-Sensitive Convolutional Neural Network for Encrypted Traffic Classification

Shiva Soleymanpour[1] · Hossein Sadr[2] · Mojdeh Nazari Soleimandarabi[2]

Accepted: 18 May 2021 / Published online: 17 June 2021

Abstract
By the rapid development of the Internet and online applications, traffic classification not only has changed to an interesting topic in the field of computer networks but also plays a key role in cyber-security and network management. Although numerous studies have been conducted in recent years, encrypted traffic classification still remains a major challenge and unbalanced data is known as one of the most important problems in this field. Even though previous researches have focused on dealing with the class imbalance problem in the pre-processing step via machine learning and specifically deep learning methods, they are still confronted with some restrictions. To this end, a new traffic classification method is presented in this paper that aims to deal with the problem of unbalanced data along the training process. The proposed method utilized a Cost-Sensitive Convolution Neural Network (CSCNN) where a cost matrix was employed to assign a cost to each misclassification based on the distribution of each class. These costs were then utilized during the training process to increase the final classification accuracy. Various experiments were carried out to explore the performance of the proposed method for the tasks of traffic classification, traffic description, and application identification. According to the obtained results, CSCNN achieved higher efficiency compared to both machine learning and deep learning based methods on the ISCX VPN-nonVPN dataset.

Keywords Traffic classification · Encrypted traffic · Deep learning · Convolutional neural network · Cost-sensitive learning

✉ Hossein Sadr
Sadr@qiau.ac.ir

Shiva Soleymanpour
Soleymanpour.sh@gmail.com

Mojdeh Nazari Soleimandarabi
Mojdeh.nazari@qiau.ac.ir

[1] Department of Computer Engineering, Ayandegan Institute of Higher Education, Tonekabon, Iran

[2] Department of Computer Engineering, Rasht Branch, Islamic Azad University, Rasht, Iran

1 Introduction

Traffic classification is generally known as a prominent concept in network security and management. Enhancing the number of users and online applications has also increased Internet traffic while about 80 % of Internet traffic belongs to peer-to-peer applications [1, 2]. The growth of bandwidth requirements and the restricted capacity of communication lines have made a remarkable need for enhancing the quality of network resource utilization. To this end, network traffic classification can help enhance network quality, licensing, accounting, and security [3].

The asymmetric architecture of today's network access links can be considered as one of the most important network classification examples that is based on this hypothesis that clients generally download more than what they upload. Furthermore, the pervasiveness of symmetric-demand applications (like peer-to-peer (P2P) applications, voice over IP (VoIP), and video call) led to alternations in clients' demands to deviate from the mentioned assumption [4]. Therefore, it is necessary to provide application-level knowledge for allocating enough resources to such applications to be able to supply a satisfactory experience for the clients. Traffic identification and classification of the various applications and services on the network are considered as the primary step for managing the networks. Furthermore, according to the rapid growth of malware, while they try to conceal their traffic to escape intrusion detection systems and firewalls, traffic classification is changed to a necessary initial step in network security systems and intrusion detection against cyber threats [5–7].

Network traffic classification methods are generally categorized into four groups of port-based, payload inspection, statistical, pattern matching, and machine learning methods [2, 8, 9]. Port-based methods employ port communications in the TCP/UDP header. In spite of the simplicity and speed of these methods, they cannot classify all protocols owing to the existence of dynamic and private ports. Furthermore, payload inspection is a very public term for acquiring the payload of a packet and it is referred to deep packet inspection. Although many techniques have been presented to increase the efficiency of payload inspection, privacy, encryption, complexity, and high processing time are still identified as their drawbacks [2].

According to the fact that those traffic classification methods that do not need access to the load capacity of packets cannot be used for specifying all protocols, some legal restrictions may be performed to prevent access to the load capacity of packets and protect user privacy. Additionally, by encrypting the load capacity, access to the load capacity will be also denied. Consequently, statistical classification methods prevent these issues by utilizing independent load capacity parameters including arrival time packet length and flow length. It must be mentioned that although statistical methods cannot provide remarkable accuracy, they are able to carry out the classification at a high speed [10, 11].

Pattern matching methods are known as another group of network traffic classification methods that have been utilized in this field for a long time. Based on the fact that they need to read the contents of packets and reading the encrypted data is hard, they are confronting with some barriers and need to overcome some problems like scalability for processing multi-GB connections and supporting large volumes of signatures [2, 12]. On the other hand, with the growth of machine learning methods, they have changed to significant methods in the field of network traffic classification. Although machine learning based methods are able to overcome some of the existing limitations and insufficiencies of the previous methods, besides requiring feature engineering, they are still confronted with two

crucial challenges [6, 13]. Firstly, they cannot support real-time classification owing to the high processing load of the large amounts of data on the network. Secondly, they are suffering from overfitting while they are confronted with unbalanced data due to their unbalanced distribution among various classes [5, 6].

Specifically, unbalanced data not only decrease the classification efficiency but also enhance the incorrect classification rate that leads to an impracticable cost to the network and causes crucial challenges to the security and management of network resources. To overcome these issues, a re-sampling method at the pre-processing step of machine learning methods has been used to balance the data. Nonetheless, loss of information and false information production are also known as the potential disadvantages of these methods [1, 14].

In recent years, deep learning methods have also achieved remarkable attention for network traffic classification owing to the fact that they do not need handcraft features and are able to extract efficient features automatically without human intervention [15, 16]. Noteworthy, class imbalance has also a negative influence on the classification efficiency of these methods. To overcome this insufficiency, a deep learning based encrypted traffic classification framework is proposed in this paper. The proposed method utilizes the Convolutional Neural Network (CNN) integrated with cost-sensitive learning to provide a classification model that aims to manage the problem of unbalanced data, which is entitled as Cost-Sensitive CNN (CSCNN). The proposed method employs a cost matrix that assigns a cost to each misclassification according to the distribution of each class. This cost is then applied to the network during the training phase. The cost matrix is created in the preprocessing step using data distribution of various classes where higher cost is assigned to the minority classes in comparison to the majority classes.

To evaluate the efficiency of the proposed method for tasks of traffic classification, application identification, and traffic description, the ISCX VPN-nonVPN [17] dataset was used in our experiments. According to the obtained results, CSCNN not only achieved higher performance in comparison to other existing methods but also was able to identify more minority class samples correctly. It can be owing to the fact that CSCNN assigns higher cost to the minority classes and lower cost to the majority classes. CSCNN then used these costs to update weight along with the training phase that made the model more sensitive to the minority classes. The advantages of the proposed traffic classification method that makes it superior compared to other classification schemes are mentioned in the following:

- The proposed method is based on deep neural networks and therefore there is no need for experts to extract features related to the network traffic.
- The proposed method can decrease the influence of unbalanced data, especially class imbalance, on the efficiency of traffic classification using a cost-sensitive matrix which yields to assigning higher cost to the minority classes and lower cost to the majority classes.
- To the best of our knowledge, deep learning has been rarely employed for the task of traffic classification. However, we did our best to compare our proposed method with two of the most famous deep learning based methods for all tasks of traffic classification, application identification, and traffic description on the ISCX VPN-nonVPN dataset.

We also compared our proposed method with machine learning based methods that have conducted their experiments on this dataset. Based on the empirical results, it can be

claimed that CSCNN has superior classification performance than both deep learning and machine learning based methods.

The rest of this paper is categorized as follows. Various kinds of network traffic classification methods besides the solutions for the problem of unbalanced data in traffic classification are completely explained in Sect. 2. The proposed method which tries to enhance the traffic classification performance on unbalanced data via training a cost-sensitive CNN is described in Sect. 3. Section 4 includes the results of the experiments and analysis while the conclusion and possible future research are mentioned in Sect. 5.

2 Related Work

An overview of the most significant traffic classification methods as well as techniques for overcoming the class imbalance problems are provided in this section. Particularly, network traffic classification methods are classified into five categories as follows [6]: (1) port-based method, (2) payload inspection, (3) pattern matching, (4) statistical, and (5) machine learning.

Port-based methods are known as the oldest and the most famous methods in this field where the classification is carried out utilizing the information in the TCP/UDP header of the packets [18]. In other words, port-based methods use the information existing in the packet headers to extract port numbers that are related to a specific application. Considering the fact that extraction is not a difficult process and port numbers are not also affected by encryption programs, these methods are not only famous for being easy and fast but also are generally used in firewalls and Access Control List (ACL) [19]. However, it is necessary to mention that besides their potential advantages, problems such as port misuse, port transmission, network address translation, and port randomness have decreased their efficiency [20], and only 30–70 % of current internet traffic can be classified using port-based classification methods [21, 22].

Payload inspection methods, commonly known as Deep Packet Inspection (DPI), employ the information existing in the application layer to perform classification where the predefined patterns like signature and regular expression of each protocol are used to distinguish protocols from each other [2]. Noteworthy, updating patterns after releasing each new protocol as well as user privacy can be considered as the main weaknesses of these methods. In this regard, Sherry et al. [23] presented a system that could solve the privacy issue by inspecting encrypted payload without decryption. However, their proposed method was only able to process HTTP secure traffic.

Following a similar line of research, Pattern matching methods are another group of network traffic classification methods that have been used in this field for a long time [24] that try to compare the packet content with a set of predefined rules in the string format. However, these methods are also confronted with particular limitations like expression limitation while they are not able to cope with complex services. To overcome these drawbacks, regular expression and dual-finite automata are commonly employed in these methods to derive suitable patterns for classification [25, 26].

Statistical methods try to overcome these problems by utilizing independent factors, such as arrival time, packet length, and flow length to perform classification [27, 28]. In other words, they are based on this assumption that the traffic of each application has some unique statistical features that can be efficiently used in classifying their underlying traffic. However, having access to a little part of statistical flow information in real-time traffic

may also jeopardize their performance. In this regard, protocol fingerprints based on the Probability Density Function (PDF) of packets inter-arrival time and normalized thresholds were presented by Crotti et al. [29] and their obtained accuracy was about 91 % for a group of protocols like HTTP, Post Office Protocol 3 (POP3), and Simple Mail Transfer Protocol (SMTP). Similarly, PDF of the packet size was also considered by Wang et al. [30]. Their proposed technique accuracy was about 87 % while it was able to determine a broader range of protocols like File Transfer Protocol (FTP), Internet Message Access Protocol (IMAP), SSH, and TELNET.

By the advancement of machine learning methods, they have attracted many researchers for the task of traffic classification due to the fact that they are able to automatically create a model from a dataset [2]. Generally, machine learning methods are divided into supervised and unsupervised techniques. Supervised techniques require labeled data to perform classification while unsupervised techniques can be performed without any prior information about the samples. In this regard, Auld et al. [31] used a Bayesian neural network to classify P2P protocols and obtained 99 % accuracy. Moore et al. [22] also used the Naïve Bayes classifier on the same application and achieved 96 % accuracy. Additionally, artificial neural networks have been used for traffic classification and acquired superior performance compared to Naïve Bayes [32]. It is worth mentioning that two of the most prominent methods that have conducted their experiments on the ISCX VPN-nonVPN traffic dataset were also based on machine learning methods. Particularly, Gil et al. [17] utilized time-related features to specify the network traffic using the C4.5 decision tree and k-nearest neighbor technique and obtained 92 % recall for classifying the six major classes of the traffic. Yamansavascilar et al. [33] also used the k-nearest neighbor technique and obtained an accuracy of 94 % for classifying 14 classes of applications on the same dataset.

Although machine learning based methods have obtained remarkable results, they are still facing obstacles including feature extraction and feature selection that are mostly performed with the help of an expert. Hence, it can be stated that feature engineering is very costly, time-consuming, and prone to human mistakes. To overcome these issues, using deep learning methods has obtained considerable attention in the field of traffic classification while they do not require any handcraft features and their highly flexible architectures can learn directly from raw data [34–36]. In this regard, Chen et al. [34] proposed a method, named Seq2Img, that utilized CNN to classify IP traffic. Based on their proposed method, stream sequences were converted into an image and CNN was employed to perform traffic classification. Wang et al. [35] proposed a method that learned the low-level spatial features of network traffic using deep CNN and Long Short-Term Memory (LSTM) network. Notably, a combination of CNN and Recurrent Neural Network (RNN) has been also recently utilized for the task of traffic classification [37, 38]. Following a similar line of research, the Datanet method [37] was proposed to efficiently manage distributed smart home networks where the classification was performed using three deep learning models including multilayer perceptron, stacked auto-encoder, and CNN. Particularly, Wang et al. [39] and Lotfollahi et al. [1] used deep neural networks for traffic classification and performed their experiments on the ISCX VPN-nonVPN traffic dataset. In this regard, Wang et al. [39] presented a method that integrated feature extraction, feature selection, and classifier into a unified end-to-end method. Lotfollahi et al. [1] also presented the Deep Packet method that leveraged a combination of stacked auto-encoder and CNN to classify the network traffic.

Noteworthy, in spite of the fact that deep learning based methods have made significant improvements in traffic classification [1, 37], the class imbalance is still known as a common problem in this field which can generally result in discriminatory and biased

classification towards majority classes. There are two prominent approaches for dealing with unbalanced data: (1) Data-level approaches that re-balance the distribution of classes in the pre-processing phase. Since data-level approaches are independent of the classification algorithms, they are flexible. Re-sampling techniques including Random Under-Sampling (RUS) and Random Over-Sampling (ROS) are two famous re-sampling techniques that are used for this aim. In RUS, some samples are randomly selected and eliminated from the majority classes. In contrast, in ROS some minority samples are randomly duplicated [40]. Information loss and fake knowledge production are known as the primary problems that RUS and ROS techniques are respectively confronted with [41]. (2) Algorithm-level approaches that alter the learning process by inserting an extra specialized mechanism into the original algorithm to determine the classifier's sensitivity toward the minority classes. Cost-sensitive is a technique that is commonly utilized to deal with class imbalance problems where variable costs are assigned to each class and then the algorithm adapts these costs during the training process to update weights. The goal of this technique is to assign a higher cost to minority class samples as well as decreasing the overall learning cost.

In this area of research, some methods have been recently presented that tried to integrate class-specific costs into deep neural networks, such as CNN [15, 42], DNN [43, 44], and auto-encoders [45]. Khan et al. [44] incorporated a cost-sensitive setting in CNN to learn feature representations and classifier parameters for both majority and minority classes. This approach can be used for both binary and multiclass problems. To the best of our knowledge, no studies have been previously conducted on using cost-sensitive techniques along with deep neural networks for the task of traffic classification aiming to overcome the class imbalance problem. It can be claimed that it is the first study that tries to deal with class imbalance problem in encrypted traffic classification by generating a cost matrix based on the class distribution and using it during the model training process and updating weight.

3 Background in Deep Neural Network

Neural networks are commonly recognized as computing systems that consist of many basic and highly interconnected elements for processing information. These networks are made of an extensive number of building blocks (neurons) that are connected to each other via links (connection) with a particular weight. Along with the training process, a large number of data samples are fed to the neural network and a learning algorithm (backpropagation) is performed to adjust the weights to obtain the desired output. Specifically, deep neural networks are also known as a particular kind of neural network that contains many hidden layers that have become more feasible in recent years due to the rapid enhancement of computational powers and accessibility of Graphical Processing Units (GPU) [46]. They have been also successfully utilized in various domains including computer vision, image processing, natural language processing, information retrieval, and especially traffic classification [8, 47]. While the focus of the proposed method of this paper is on using CNN for traffic classification, more details about this network and how it is used for traffic classification are provided in the following section.

CNN is one of the typical deep neural networks that can perform automatic feature extraction using layers comprised of convolutional operations and is generally suitable for sequential data like language [1, 48, 49]. The main concept of CNN is to achieve local

features from the input at higher layers and then combine them into more complex features at lower layers. In this regard, CNN leverages the convolutional layer as a basic building block which takes $N \times N$ square neurons and a filter of $w$with the size of $m \times m$ as input. By applying the convolutional operation on these two matrices, the output features of each layer c^l with the size of $(N - m + 1) \times (N - m + 1)$ are obtained. Thereafter, f is applied as the activation function to learn more complex features (Eq. 1). Pooling operation is also applied over the obtained feature maps to aggregate multiple low-level features and reduce computational costs [8].

$$c_{ij}^{l} = f\left(\sum_{a=0}^{m-1}\sum_{b=0}^{m-1} w_{ab} c_{(i+a)(j+b)}^{l-1}\right) \tag{1}$$

Due to the fact that network traffic is essentially sequential data and the structure of byte, packet, session, and the whole traffic are very similar to the structure of character, word, sentence, and the whole document in the natural language processing, CNN can be also used for encrypted traffic classification. In this regard, consider $X_i \in \mathbb{R}$ is a k-dimensional vector that corresponds to the ith traffic byte in the session or flow. Therefore, $X_{1:n} = x_1 \oplus x_2 \oplus \cdots \oplus x_n$ represents a flow of length n where $\oplus$is the concatenation operation. Generally, $x_{i:i+j}$ presents the concatenation of traffic bytes $x_1, x_{i+1}, \ldots, x_{i+j}$. Then, the convolutional operation with the filter $w \in \mathbb{R}$ is applied to a window of h traffic bytes to generate a new feature $c_i \in \mathbb{R}$(Eq. 2).

$$c_i = f(w \circ x_{i:i+h-1} + b) \tag{2}$$

Here b is the bias term, f is the activation function (*ReLU*), and$\circ$ refers to the dot product between the convolutional filter and traffic submatrix. The filter is applied to each possible window of traffic bytes $\{x_{1:h}, x_{2:h+1}, \ldots, x_{n-h+1:n}\}$ to generate feature mapsc_i (Eq. 3).

$$c_i = \left\{c_1, c_2, \ldots, c_{n-h+1}\right\} \tag{3}$$

Then, a max-pooling operation is applied over the obtained feature maps to obtain the maximum value as the next feature (Eq. 4)

$$\hat{c} = \max\left(c_i\right) \tag{4}$$

The achieved features are passed to a fully connected *SoftMax* to determine the probability distribution of the input session or flow (Eq. 5).

$$y = Softmax(W^{(s)}\hat{c} + b) \tag{5}$$

Overall, that it can be stated that CNN can be an ideal choice for network traffic classification because it is able to capture spatial dependencies between adjacent bytes in network packets which results in finding discriminative patterns for every class of protocols/applications that can lead to the accurate traffic classification.

4 Methodology

According to the fact that unbalanced data have a considerable influence on the efficiency of CNN and can cause over-fitting during the training process [42], different methods have been proposed to fill these lacunae. Existing methods [1, 35] generally utilized re-sampling techniques that needed expert knowledge to specify the majority and minority classes. Besides being time-consuming and costly, this technique not only yields to the removal of many data patterns but also results in false data patterns generation. In this regard, a Cost-Sensitive CNN- (CSCNN) method for the task of encrypted traffic classification is proposed in this paper. The schematic representation of the proposed method is illustrated in Fig. 1.

As it is obvious, the proposed method includes two separate phases. Accordingly, the network traffic data are cleaned to become suitable as the input of the neural network in the first phases. while the second phase consists of the proposed CSCNN method. More details about the used dataset and the proposed method are provided in the following.

4.1 Dataset

Generally, self-collected traffic or security companies' private traffic have been used for the evaluation of encrypted traffic classification methods that resulted in incompatibility between their obtained results. In other words, public datasets have been rarely employed for evaluation in this field and while classical machine learning requires handcraft features as input, most of the existing public datasets are feature datasets rather than traffic datasets. However, Gil et al. [17] generated a public dataset for the task of encrypted

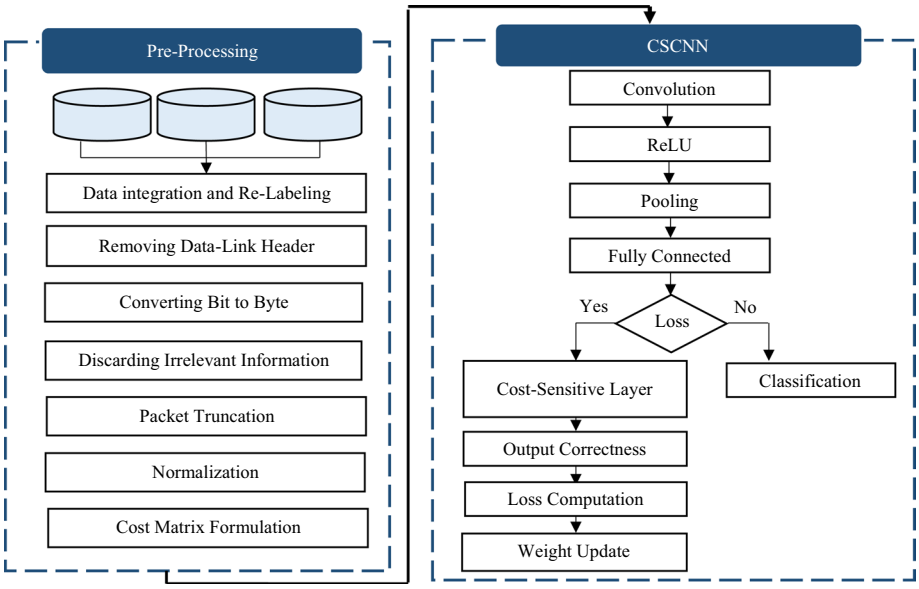

Fig. 1 Schematic representation of the proposed method

traffic classification (ISCX VPN-nonVPN dataset) that includes captured traffic of different applications in *pcap* format files. Each file was labeled according to the application produced the packets (e.g., Skype, and Hangouts) along with the activity the application was engaged during the capture session (e.g., voice call, chat, file transfer, or video call). The dataset also contains packets captured over Virtual Private Network (VPN) sessions. Like non-VPN traffic, VPN traffic is captured for different applications, such as Skype, while performing different activities, like voice calls, video calls, and chat.

Having the above-mentioned issues in our mind, the "ISCX VPN-nonVPN" dataset was used in our experiments in order to be able to provide a fair comparison between our proposed method and other existing methods for the task of encrypted traffic classification.

4.2 Pre-Processing

While the "ISCX VPN-nonVPN" dataset file format is not suitable to be used as the input of the proposed CSCNN method, the raw traffic of this dataset must be pre-processed to generate the required format. The pre-processing phase contains seven fundamental steps that are explained in the following:

(1) *Data integration and re-labeling*: All *pcap* files are merged to form a single dataset in the integration phase. While *pcap* files are labeled according to their applications, they must be re-labeled for application identification and traffic description. Re-labeling respectively resulted in 17 and 14 classes for application identification and traffic description. More details about re-labeled classes are reposted in Table 1.
(2) *Converting bit to byte*: The values in the dataset are stored as bits between 0 and ff, including eight bits. To reduce the input size, the data is first converted to byte format and then is converted to a value between 0 and 255.
(3) *Discarding irrelevant information*: While the dataset is collected in a real-world simulation, it consists of several useless packets that are not suitable for modeling and must

Table 1 Number of samples per class: (A) Traffic description (B) Application Identification

A		B	
Class name	Size	Application	Size
Browsing	2500	AIM chat	5k
Chat	890	Email	28k
Email	249	Facebook	2502k
File transfer	1018	FTPS	7872k
Streaming	482	Gmail	12k
Torrent	1000	Hangouts	3766k
VoIP	2826	ICQ	7k
Vpn: Browsing	2500	Netflix	299k
Vpn: chat	1196	SCP	448k
Vpn: File transfer	1932	SFTP	418k
Vpn: Email	491	Skype	418k
Vpn: Streaming	475	Spotify	2872k
Vpn: Torrent	928	Torrent	40k
Vpn: VoIP	2271	Tor	70k
		VoIP buster	202k
		Vimeo	842k
		YouTube	146k

be removed. As matter of fact, the ISCX VPN-nonVPN dataset consists of some TCP segments with SYN, ACK, or FIN flag sets that are essential for a three-way handshaking procedure or establishing or finishing a connection. In contrast, these segments are not carrying any suitable information about the application that had generated them and must be discarded. Moreover, Domain Name Service (DNS) segments that are not relevant to application identification and traffic classification must be also removed from the dataset.

(4) *Packet truncation*: Due to the fact that neural networks need fixed-length inputs and the packet length varies a lot through the dataset, the length of packets must be united. To this end, the size of all packets is fixed to 1480 by cutting or zero-padding.

(5) *Normalization*: To achieve higher efficiency, all the packet bytes are divided by 255 while the input values are in the range [0–1].

(6) *Cost matrix formulation*: The cost matrix γ is formulated in the last level of the pre-processing phase and is then applied to the output of the last layer of CNN to alter the network's weights based on various costs. While in CNN higher score is commonly assigned to the output class, the goal of the cost matrix is to assign the maximum cost to the minority classes while lower costs are assigned to other classes (majority ones). In a cost matrix, the diagonal of the matrix is known as the utility vector. This vector presents the correct classification and is set to zero. For other classifications, all costs are non-negative, i.e. $\gamma_{i,j} > 0$. An example of a cost matrix for a dataset with four classes is presented in Table 2 where a 4×4 matrix is generated when all the cells of the matrix are larger than zero, except those in the diagonal row that are always set to zero.

(6) *Removing data-link header*: According to the fact that the ISCX VPN-nonVPN dataset is captured at the data-link layer and the data-link header includes physical link information like a Media Access Address (MAC) that is also necessary for transmitting frames over the network but is not essential and informative for application identification or traffic description, the Ethernet header must be eliminated in the pre-processing phase

Consequently, it can be stated that if the algorithm accurately classifies the sample, there is no cost. In other respect, the proposed algorithm assigns a cost to misclassification based on the distribution of the corresponding classes. The cost of each class is also computed using the following equation (Eq. 6).

Table 2 Traffic description performance

		Classes			
		C_1	C_2	C_3	C_4
Classes	C_1	0	$Cost_{c1\to c2}$	$Cost_{c1\to c3}$	$Cost_{c1\to c4}$
	C_2	$Cost_{c2\to c1}$	0	$Cost_{c2\to c3}$	$Cost_{c2\to c4}$
	C_3	$Cost_{c3\to c1}$	$Cost_{c3\to c2}$	**0**	$Cost_{c3\to c4}$
	C_4	$Cost_{c4\to c1}$	$Cost_{c4\to c2}$	$Cost_{c4\to c3}$	0

$$\text{Cost}_{ci \to cj} = \begin{cases} 1 - \dfrac{N_{ci}}{N_{ci} + N_{cj}}, & \text{if } \dfrac{N_{ci}}{N_{cj}} > 1 \\ \dfrac{N_{ci}}{N_{ci} + N_{cj}}, & \text{Otherwise} \end{cases} \tag{6}$$

4.3 Cost-Sensitive Convolutional Neural Network (CSCNN)

Based on previous studies, it can be concluded that using cost-sensitive learning along with the training of a neural network can result in higher efficiency compared to data-level methods. As a matter of fact, cost-sensitive learning is a subfield of machine learning which considers the cost of prediction errors along with the training of a model. It is also closely related to the field of imbalanced learning which involves explicitly defining and using cost during the training process. In this regard, a Cost-Sensitive CNN (CSCNN) is proposed in this paper that tries to learn appropriate features for the minority and majority classes automatically to improve the efficiency of traffic classification.

The main notion behind cost-sensitive learning is to make a priority on minority class instances while facing misclassification. Accordingly, a cost is assigned to each misclassification type while misclassifications of the minority classes obtain higher values compared to misclassifications of the majority classes. Once a misclassification happens, the cost function is then activated and attempts to enhance the cost value by applying the corresponding cost that was pre-defined by the user or automatically assigned by a technique. Therefore, cost-sensitive learning results in a higher cost value when a minority class instance is misclassified in comparison to the time when a majority class instance is misclassified. Considering this cost along with updating the parameters of the neural networks, the training process becomes more sensitive to minority classes. Despite previous methods that utilized a user-defined matrix, the proposed CSCNN automatically adjusts the cost of each classification using the data distribution. The details of the

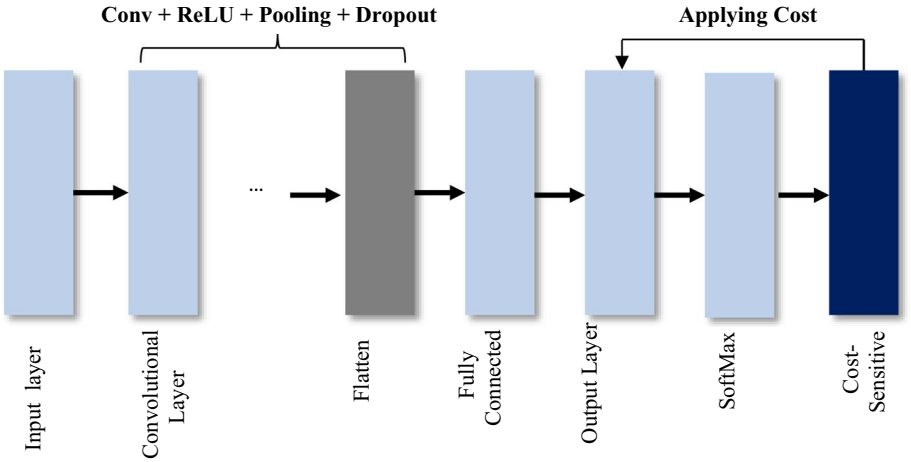

Fig. 2 Cost-Sensitive CNN algorithm

proposed CSCNN are presented in Fig. 2 which includes three primary steps: (1) forming a cost matrix, (2) learning features using CNN, and (3) a cost-sensitive function.

Noteworthy, the goal of the proposed method is to decrease the influence of unbalanced data on the efficiency of encrypted network traffic classification as well as focusing on learning data that have uneven penalties or costs when making a prediction. To this end, a cost matrix is formed by computing the distribution of samples of all classes in the first step. Thereafter, the data is fed to CNN to carry out the classification in the second step which can be considered as an end-to-end strategy that is able to directly learn the nonlinear relationship between traffic input and expected output label rather than diving the problem into subproblems. It is worth mentioning that a two-layers CNN where a convolution layer is followed by a *ReLU* activation function and a max-pooling layer is also utilized. *SoftMax* classifier, which is a multi-class version of logistic regression, is finally used to specify the output classes.

Noteworthy, due to the fact that the compromise is to train on the training dataset but to stop training at the point when performance on a validation dataset starts to degrade, an early stop technique is also utilized to intercept over-fitting. Accordingly, If the value of the loss function of the validation set does not change for several iterations, the training process stops. To standardize the input and stabilize the learning process besides reducing the number of training epochs and accelerating the learning process, batch normalization technique is also used during the training process.

Thereafter, the actual and predicted classes are first specified using a cost-sensitive layer (third step) after each misclassification. The misclassification cost is then determined utilizing the cost matrix to be used for output modification. Ultimately, the specified cost is then applied to the outputs of actual and predicted classes. In summary, the classification process is determined as follows:

(1) The first convolutional layer makes use of a set of learnable filters where the input data is processed with 8 filters (filter size= [1, 3]). Each filter moves 1 step after one convolutional operation. Convolving the same filters at every position in the input matrix allows the features to be extracted automatically.
(2) Features obtained from the convolution layer are fed to *ReLU* as an activation function to learn complex patterns in the data (Eq. 7).

$$ReLU(x) = max[0, x] \tag{7}$$

(3) After applying *ReLU* function, the results are then processed through the pooling layer to reduce the dimension of the features. The pooling layer operates upon each feature map separately to create a new set of the same number of pooled feature maps. Max-pooling is used in the proposed method which processes a [1, 2] input as follows (Eq. 8). The max-pooling has a step size of 1.

$$maxpooling\left[x_1, x_2, x_3, x_4\right] = max(x_1, x_2, x_3, x_4) \tag{8}$$

(4) Three aforementioned layers (i.e., convolutional layer, *ReLU*, and max-pooling) are then added to the network with the same settings described in layers 1, 2, and 3.
(5) These features are then passed to a fully connected *SoftMax* layer whose output is the probability (Eq. 9).

$$f_\theta(x) = \frac{1}{\sum_{j=1}^{C} e^{y_j}} \left[e^{y_1} e^{y_2} \ldots e^{y_C}\right] = \left[p(y_i = 1|x_i) p(y_i = 2|x_i) \ldots p(y_i = C|x_i)\right] \tag{9}$$

A cost-sensitive strategy is then utilized to address the class imbalance problem along with the feature learning process via improving the efficiency of the cost function where weight and bias parameters are learned by the *SoftMax* classifier to minimize cost. In fact, the purpose of a cost-sensitive strategy is to punish all kinds of classification errors based on certain costs. The basic idea behind this strategy is that the larger output values in the output layers yield to higher probability in comparison to the smaller output values in the *SoftMax* layer. Consequently, CSCNN tries to decrease the predicted class output value and enhance the actual class output using Eqs. (10) and (11).

$$y_k = y_P - \gamma_{i,k} \times y_P \tag{10}$$

$$y_i = y_A + \gamma_{i,k} \times y_A \tag{11}$$

Here the terms y_P and y_A respectively refer to the predicted class and the actual class outputs. The y_i and y_k values also respectively present the new outputs for actual and predicted classes. The cross-entropy cost function is then altered and a new cost function is introduced. The new function obtains y and p values as inputs and returns a loss value for each class. After modifying the output of the two actual and predicted classes, the probability values are again calculated utilizing the *SoftMax* function (that are known as p_k and p_i respectively for the predicted (y_k) and actual (y_i) classes). The new loss value for the predicted and the actual classes are computed using Eqs. (12) and (13) respectively.

$$-\left(yloglog\left(p_k\right) + (1-y)loglog\left(1-p_k\right)\right) \tag{12}$$

$$-\left(yloglog\left(p_i\right) + (1-y)loglog\left(1-p_i\right)\right) \tag{13}$$

Since the value of y is equal to1 for the predicted class and is zero for the actual class, the predicted class activates the first part of the equation, i.e. $-\log(p)$. On the contrary, the actual class activates the second part of the equation, i.e. $-\log(1-p)$. Since the probability value of p for the predicted class is decreased (by enhancing the value in the output layer, y_k), the $-\log(p)$ value becomes larger. Furthermore, as the probability value (p) of the actual class is enhanced (by reducing the output y_i), the p value in the $-\log(1-p-1)$ is reduced. Consequently, the outputs of the loss function are enhanced for both classes that leads to an enhancement in the overall loss cost of various classes.

In general, the proposed CSCNN aims to decrease the neural network cost by imposing punishments on various misclassifications. These costs are specified utilizing the classes' distribution in such a way that classes with fewer samples obtain higher costs and the majority classes obtain lower costs aiming to train a network that is more sensitive to minority classes.

5 Results and Discussion

In order to prove the efficiency of our proposed method, we decided to carry out various experiments in terms of traffic classification, traffic description, and application identification. It is worth mentioning that all implementations were conducted using Python as the programming language, Anaconda as a library for implementing deep neural networks, and Tensorflow as its backend. Furthermore, the learning rate and the number of epochs were respectively set to 0.1 and 100. The size of convolutional filters and pooling filter were respectively set to 1×12 and 1×3. To prevent overfitting, the dropout layer with a masking probability of 0.4 was applied for regularization. A fully connected network with two hidden layers was also implemented to perform the final classification. Stochastic Gradient Descent (SGD) and cross-entropy were respectively utilized as optimizer and loss function. To perform training, 90 % of data was randomly selected as a training set and the remained 10 % of data was used as a test set. Implementations were conducted on a system with an Intel Xeon 2 E5-2620 2.0 GHz processor and 8 GB of RAM running windows server 2008.

To evaluate the proposed method, five commonly used metrics, known as Accuracy, Recall, Precision, F1 Score, and False Alarm Rate (FAR) were employed in our experiments and their equations are provided in the following. Where TP, TN, FP, and FN respectively refer to the true positive, true negative, false positive, and false negative.

$$\text{Accuracy} = \frac{TP + TN}{FP + FN + TP + TN} \tag{14}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{16}$$

$$\text{F1 Score} = \frac{2 * \text{Recall} * \text{Percision}}{\text{Recall} + \text{Percision}} \tag{17}$$

$$\text{FAR} = \frac{FP}{FP + TN} \tag{18}$$

As previously mentioned, deep neural networks have been rarely employed for the task of traffic classification and most of the existing methods also have used a particular self-collected traffic dataset in their experiments. Moreover, the efficiency of deep learning based methods is highly related to the used hardware. In this regard, comparing the proposed method with other state of the art is very complicated and confusing. However, in order to prove the superior performance of the proposed method, we compared it with two of the most prominent methods in this field, namely Deep Packet [1] and Datanet [37] in terms of traffic classification, traffic description, and application identification on ISCX dataset. Noteworthy, the proposed method is also compared with a machine learning based method while their results were taken from their original papers. More details about the obtained results are provided in the following.

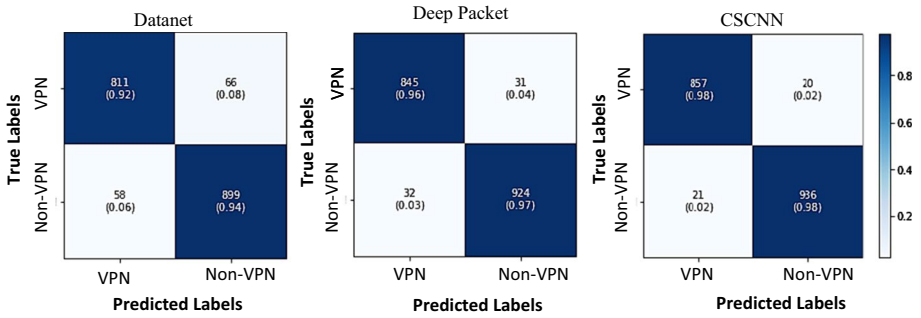

Fig. 3 Confusion matrices of classification for VPN and Non-VPN classes

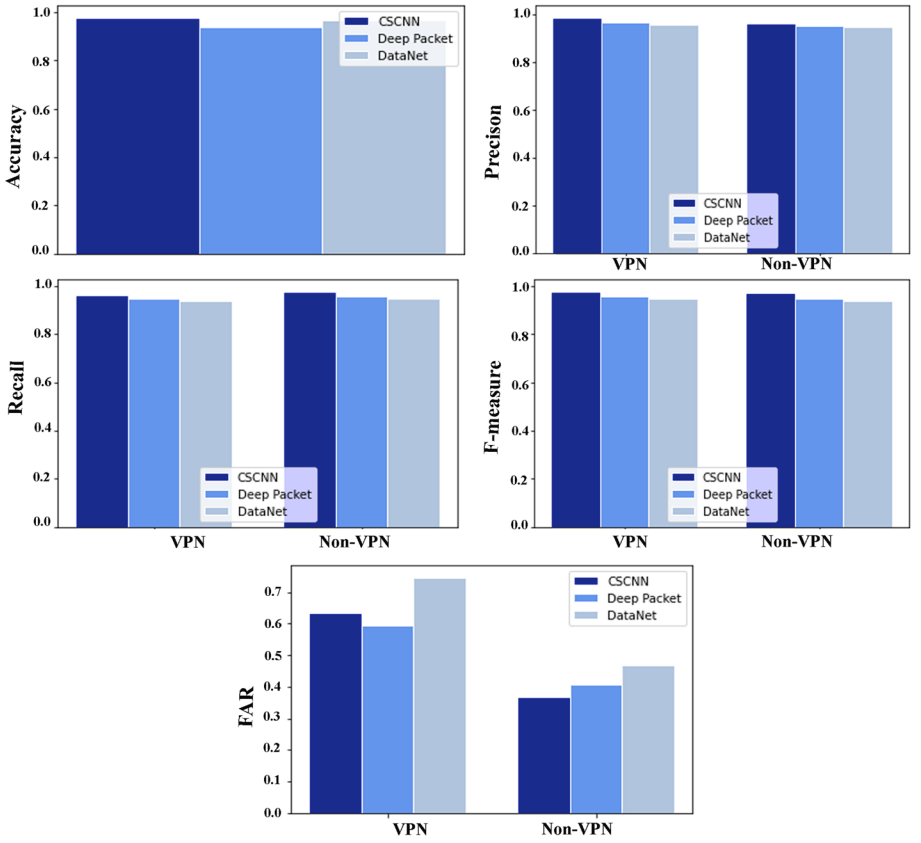

Fig. 4 Comparison of the two-class classification performance of CSCNN, Deep Packet, and Datanet methods

5.1 Classification Results

The performance of the proposed method in terms of classification between VPN and Non-VPN classes compared to the Deep Packet [1] and Datanet [37] methods is investigated in this section. The Confusion matrices for all three methods are illustrated in Fig. 3. As can be seen, the proposed CSCNN method is able to detect VPN and Non-VPN with 98 % accuracy while only 0.02 % of traffic was misclassified. On the other hand, Deep Packet [1] and Datanet [37] respectively obtained the accuracy of 97 and 94 % for Non-VPN classification and 96 and 92 % for VPN classification. Apparently, the proposed CSCNN method has superior performance for the task of classification.

In order to provide more comparison, the results obtained from the confusion matrices of Fig. 3 are illustrated in Fig. 4 based on five metrics of accuracy, recall, f-measure, precision, and false alarm rate. As can be seen, although CSCNN has slightly higher performance than Deep Packet [1] and Datanet [37] based on four measures of accuracy, recall, f-measure, and precision, their performances can be also considered relatively identical which can be attributed to the balanced distribution of each class data while each of the

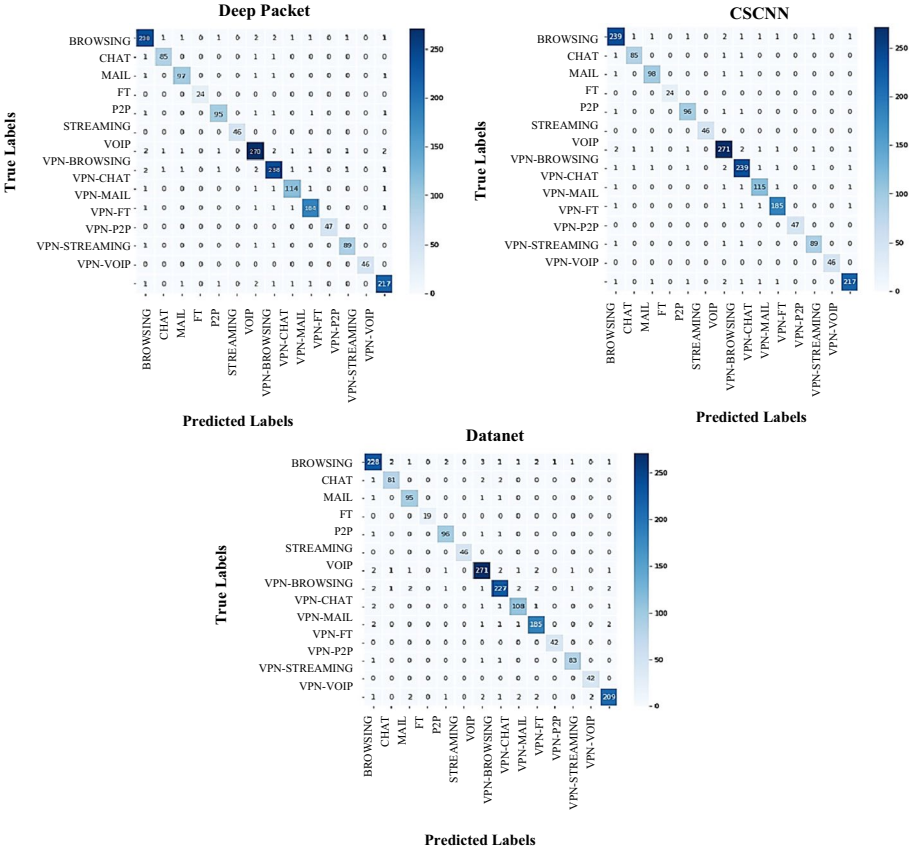

Fig. 5 Confusion matrices of traffic description (14 classes)

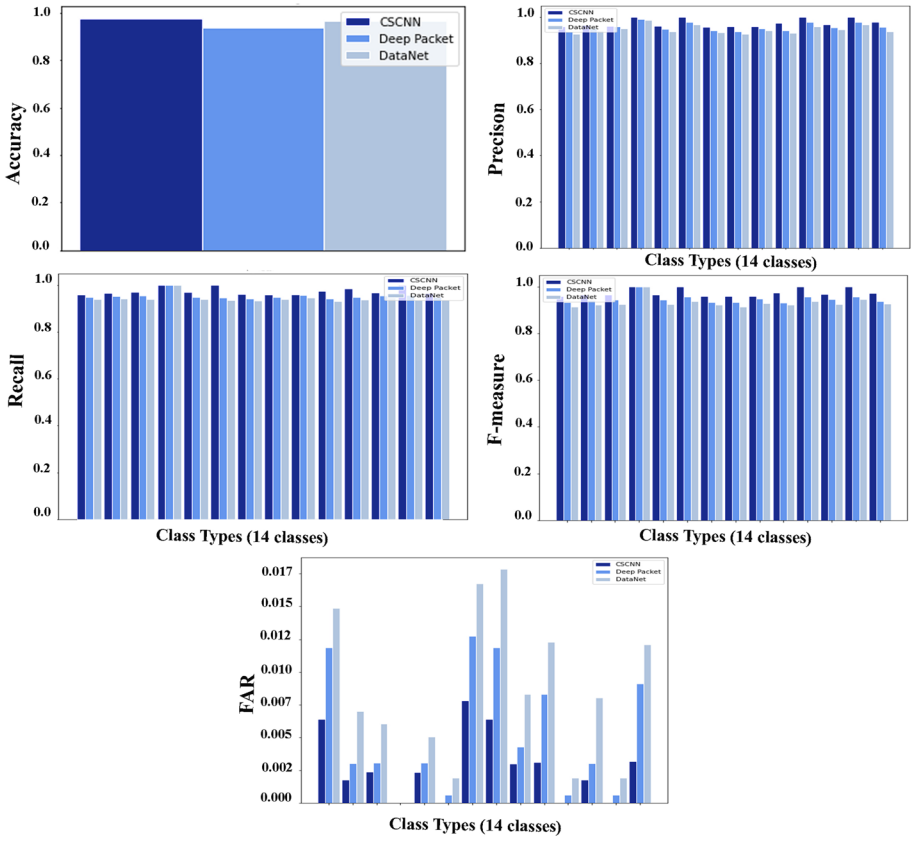

Fig. 6 Comparison of the traffic description performance of CSCNN, Deep Packet, and Datanet methods

VPN and Non-VPN classes includes about 50 % of the total data. Moreover, the false alarm rate of CSCNN is lower on Non-VPN traffic compared to two other mentioned methods.

5.2 Traffic Description Results

Traffic description is another factor that can be considered for evaluating the performance of the proposed method. In this regard, the confusion matrices of CSCNN, Deep Packet [1], and Datanet [37] are illustrated in Fig. 5 where the rows of the confusion matrices correspond to the actual class and the columns are related to the predicted labels and therefore the matrices are row normalized. Noteworthy, the darker color of the diagonal elements states that the method can classify each application with minor confusion because they present the accurately classified results. By carefully observing the confusion matrices of these three methods, it is clear that the number of misclassification of the CSCNN is lower compared to the other two methods that can clearly prove the efficiency of our proposed method.

In order to provide more comparison, the results obtained from the confusion matrices of Fig. 5 are illustrated in Fig. 6 based on five metrics of accuracy, recall, f-measure, precision, and false alarm rate. As it can be seen, CSCNN presented better results compared

Table 3 Traffic description performance of CSCNN, Deep Packet, and Datanet method

	CSCNN					Deep Packet					Datanet				
Category	Acc	Rec	Pre	F1-sc	FAR	Acc	Rec	Pre	F1-sc	FAR	Acc	Rec	Pre	F1-sc	FAR
Browsing	97.7	96	96	96	0.06	96.3	95.6	95.6	95.6	0.07	96.7	93.9	92.8	91.3	0.015
Chat		96.6	96.6	96.6	0.02		95.8	95.4	96	0.02		94.3	95.7	94.6	0.007
Mail		97	96.1	96.6	0.02		95.1	95.4	94.3	0.02		94.0	94.9	92.3	0.006
FT		100	100	100	0		100	100	100	0		100	98.7	92.5	0.000
P2P		97	96	96.5	0.02		95	96	95.5	0.02		93.9	93.8	92.4	0.005
Streaming		100	100	100	0		100	100	100	0		93.6	96.8	93.7	0.002
VOIP		96.1	95.8	95.9	0.08		95.7	95.7	95.7	0.08		93.2	94.7	93.9	0.017
VPN-Browsing		96	96	96	0.06		94.2	95.6	94.6	0.07		93.9	92.8	93.2	0.018
VPN-Chat		95.8	95.8	95.8	0.03		94.3	94.8	95.2	0.03		94.7	94.1	92.9	0.008
VPN-Mail		97.4	97.4	97.4	0.03		97.4	96.3	96.8	0.04		93.2	93.1	92.2	0.012
VPN-FT		98.6	100	100	0		100	100	100	0		93.9	95.8	93.7	0.002
VPN-P2P		96.7	96.7	96.7	0.02		96.7	96.7	96.7	0.02		94.5	95.2	94.8	0.008
VPN-Streaming		100	100	100	0		99.1	98.7	99	0.01		93.6	96.8	94.7	0.002
VPN-VOIP		96.4	97.7	97.1	0.03		95.1	96.4	95.6	0.05		93.8	94.1	93.9	0.012
Average		97.4	97.4	97.4	0.026		96.7	96.9	96.7	0.03		94.3	94.9	**93.2**	0.008

****Acc**, Accuracy; *Rec*, Recall; *Pre*, Precision; *F1-sc*, F1-Score; *AR*, False Alarm Rate

to the other two methods. The better performance of CSCNN is clearer in false alarm rate comparison that can be due to the higher error rate for the minority class samples leads to higher values compared to Deep Packet [1] and Datanet [37].

Moreover, this value is higher for Chat، File transfer، Streaming, and VPN: Streaming compared to other classes. The accuracy of the CSCNN is about 96.9% while this value is equal to 94.3 and 96.71% for Deep Packet [1] and Datanet [37] respectively. Therefore, it can be concluded that the accuracies of all three methods are relatively low which can be due to the fact that this measure of classification cannot consider all the classes. In this regard, it can be claimed that accuracy is not an appropriate measure for evaluating unbalanced data classification. In contrast, recall and precision have better efficiency in evaluating the classification of unbalanced data. Notably, the recall and precision values are about 97.4% for CSCNN while these values are lower for Deep Packet [1] and Datanet [37]. The comparison of the CSCNN, Deep Packet [1], and Datanet [37] methods for traffic description according to five metrics of accuracy, recall, f-measure, precision, and false alarm rate is also presented in Table 3. The higher performance of the CSCNN in all evaluation metrics proves that it has entirely extracted and learned the discriminative features from the training set and can successfully classify traffic description.

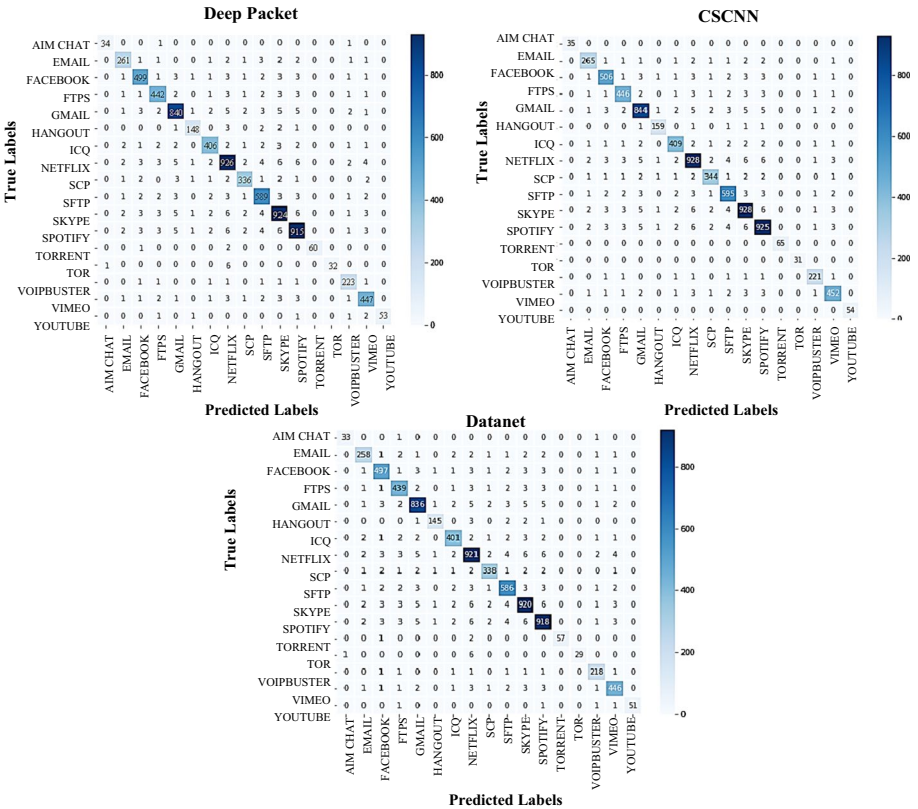

Fig. 7 Confusion matrices of application identification (17 classes)

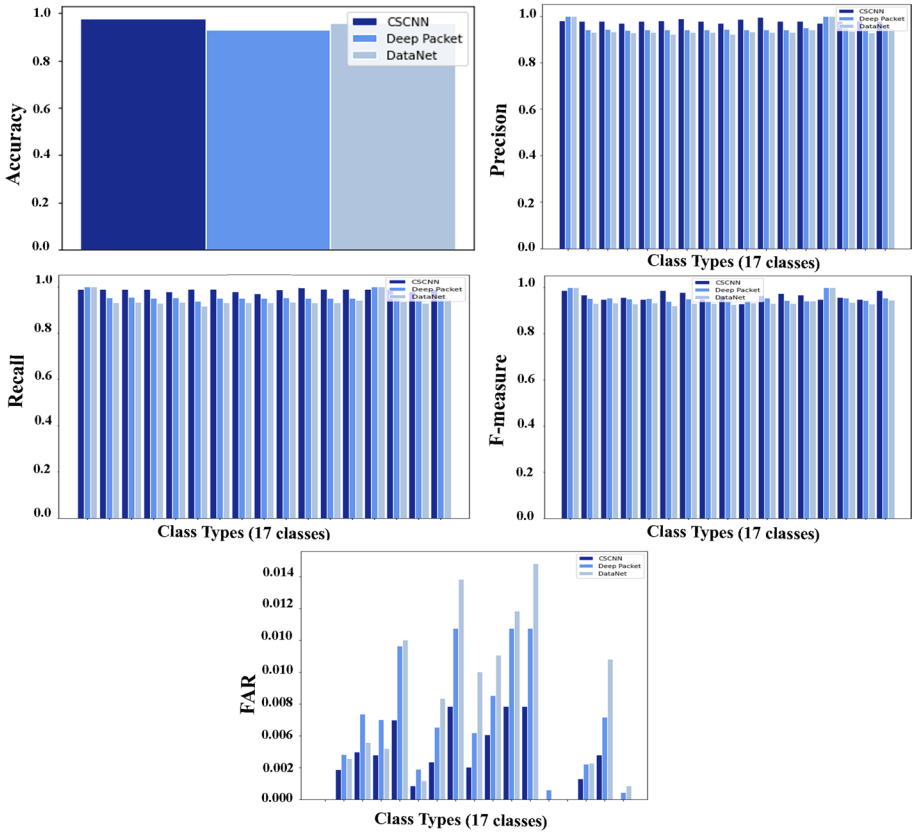

Fig. 8 Comparison of the application identification performance of CSCNN, Deep Packet, and Datanet methods

5.3 Application Identification Results

Application identification is another factor that was used to evaluate the performance of the proposed method. In this regard, the confusion matrices of CSNN, Deep Packet [1], and Datanet [37] are illustrated in Fig. 7 where the rows of the confusion matrices correspond to the actual class and the columns are related to the predicted labels and therefore the matrices are row normalized. As it is clear, CSCNN can correctly classify Youtube, Tor, Torrent, and AIM Chat classes. Moreover, by carefully observing the confusion matrices of these three methods, it is obvious that CSCNN can classify samples more efficiently while its misclassification rate is lower compared to the other two methods for application identification.

In order to provide more comparison, the results obtained from the confusion matrices of Fig. 7 are illustrated in Fig. 8; Table 4 based on five metrics of accuracy, recall, f-measure, precision, and false alarm rate. As it can be seen, CSCNN presented better results compared to the other two methods. The accuracy of the CSCNN is about 97.9 % while this value is equal to 96.2 and 96.1 % for Deep Packet [1] and Datanet [37] respectively and the recall average is about 98.6, 96.4 and 93.9 % for CSCNN, Deep Packet

Table 4 Application identification performance of CSCNN, Deep Packet, and Datanet method

	CSCNN					Deep Packet					Datanet				
Category	Acc	Rec	Pre	F1-sc	FAR	Acc	Rec	Pre	F1-sc	FAR	Acc	Rec	Pre	F1-sc	FAR
AIM_chat	97.9	98.9	97.9	98.7	0.065	96.2	97.3	97.3	96.8	0.089	96.1	100	100	100	0.000
Email		98.8	97.8	96.7	0.028		96.6	96.6	95.7	0.038		93.1	93.2	93.1	0.003
Facebook		98.8	97.8	94.7	0.005		96.5	97.5	94	0.008		93.3	94.7	94.0	0.004
FTPS		98.9	96.9	95.6	0.002		96.3	95.3	93.1	0.002		92.9	93.1	93.1	0.003
Gmail		97.7	97.7	94.7	0.019		95.9	96.9	93	0.024		93.3	93.2	93.2	0.010
Hangouts		98.9	97.9	98.7	0.003		97.3	97.3	97.7	0.001		91.6	92.2	91.9	0.001
ICQ		98.8	98.8	97.7	0.085		96.6	95.6	95.7	0.092		92.9	93.5	92.3	0.006
Netflix		97.8	97.8	96.7	0.007		94.5	94.5	95.7	0.011		93.0	93.1	93.1	0.014
SCP		96.9	96.9	96.6	0.008		94.3	92.3	96.6	0.012		93.0	92.2	92.6	0.008
SFTP		98.7	98.7	92.7	0.006		96.9	95.9	92.7	0.01		93.2	93.6	93.4	0.009
Skype		99.5	99.5	96.6	0.02		97.9	97.9	96.6	0.041		93.0	93.1	93.1	0.012
Spotify		98.9	97.8	97.3	0.01		97.3	97.5	97.3	0.02		92.6	94.2	93.6	0.015
Torrent		98.8	97.8	96.7	0.015		96.6	95.3	93.1	0.021		94.0	95.4	94.7	0.000
Tor		98.8	96.9	94.7	0.014		96.5	96.9	93	0.013		100	100	100	0.000
Voipbuster		98.9	97.7	95.6	0.001		96.3	97.3	97.7	0.001		93.4	95.3	94.5	0.002
Vimeo		97.7	97.9	94.7	0.004		95.9	95.6	95.7	0.005		92.8	93.6	92.1	0.009
Youtube		98.8	97.8	98.7	0.011		96.6	96.9	95.7	0.021		94.5	94.7	94.6	0.001
Average		98.6	97.9	96.3	0.017		96.4	96.3	95.3	0.024		93.9	94.4	94.1	0.006

*******Acc* = Accuracy, *Rec* = Recall, *Pre* = Precision, *F1-sc* = F1-Score, *FAR* = False Alarm Rate

[1], and Datanet [37] respectively. The higher performance of the CSCNN in all evaluation metrics proves that it has entirely extracted and learned the discriminative features from the training set and can successfully distinguish each application.

Application identification is another factor that was used to evaluate the performance of the proposed method. In this regard, the confusion matrices of CSNN, Deep Packet [1], and Datanet [37] are illustrated in Fig. 7 where the rows of the confusion matrices correspond to the actual class and the columns are related to the predicted labels and therefore the matrices are row normalized. As it is clear, CSCNN can correctly classify Youtube, Tor, Torrent, and AIM Chat classes. Moreover, by carefully observing the confusion matrices of these three methods, it is obvious that CSCNN can classify samples more efficiently while its misclassification rate is lower compared to the other two methods for application identification.

In order to provide more comparison, the results obtained from the confusion matrices of Fig. 7 are illustrated in Fig. 8; Table 4 based on five metrics of accuracy, recall, f-measure, precision, and false alarm rate. As it can be seen, CSCNN presented better results compared to the other two methods. The accuracy of the CSCNN is about 97.9 % while this value is equal to 96.2 and 96.1 % for Deep Packet [1] and Datanet [37] respectively and the recall average is about 98.6, 96.4, and 93.9 % for CSCNN, Deep Packet [1], and Datanet [37] respectively. The higher performance of the CSCNN in all evaluation metrics proves that it has entirely extracted and learned the discriminative features from the training set and can successfully distinguish each application.

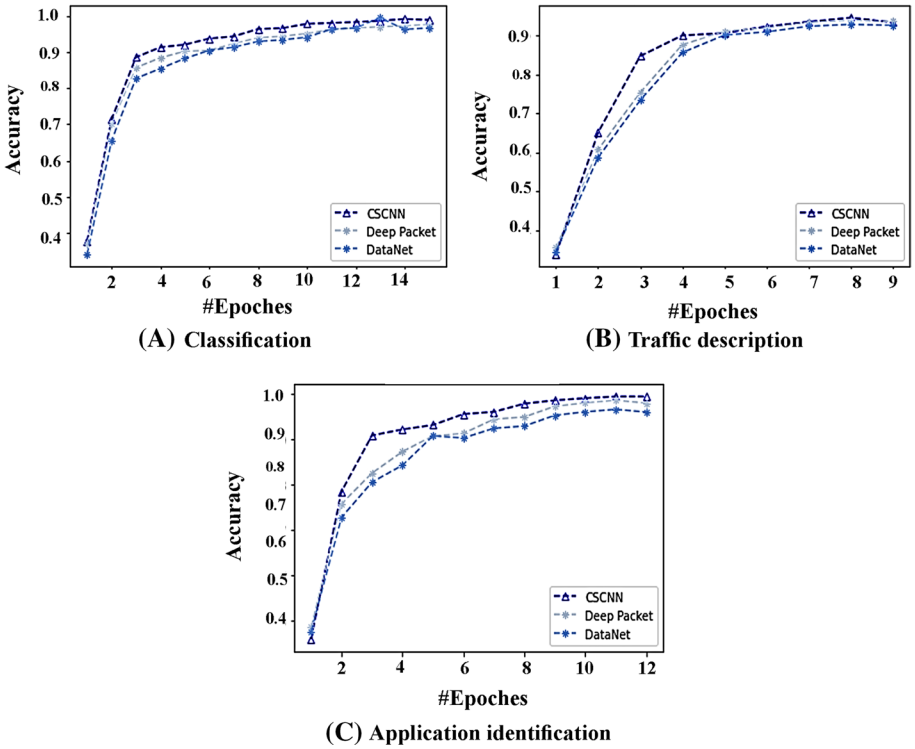

Fig. 9 Training accuracy comparison based on the number of epochs

5.4 Training Time Analysis

Considering the fact that the training time of deep neural networks is highly related to the hardware that they are implemented on, namely modern GPUs can significantly reduce the training time, it cannot be considered as a fair measure for comparing the efficiency of deep learning based methods and it has been rarely explored as a metric for evaluation. However, in order to provide an analysis of the time complexity of our proposed method, we decided to plot the training accuracy of our proposed method (CSCNN) compared to Deep Packet [1] and Datanet [37] that is illustrated in Fig. 9. As it is clear, the classification accuracy of these three methods based on the number of epochs is very close to each other in two-class classification (Fig. 9.A) which can be due to the fact that the data are unbalanced and there is a balanced rate for both classes. In the case of traffic description and application identification, it is obvious that not only CSCNN has higher accuracy but also it obtained the maximum accuracy only after four epochs and therefore it was converged faster compared to the other two methods.

Training time per epoch is another factor that can be considered for time analysis. In this regard, the runtime of these three methods, namely CSCNN, Deep Packet [1], and Datanet [37], based on the number of epochs for the tasks of traffic description and application identification is depicted in Fig. 10. As it is clear, Deep Packet [1] and Datanet [37] required training time is very close to each other while CSCNN requires more time for training in each epoch. After all, although CSCNN requires more training time per each epoch, it is converged in a lower number of epochs (about after four epochs) compared to the other two methods. Therefore, it can be concluded that the difference between their training time is not very critical. However, it is necessary to mention that choosing an optimal model for traffic classification task is not possible while the definition of "optimal" is not well-defined and there is always a tradeoff between the model complexity (training and test speed) and its performance.

5.5 Discussion

As previously mentioned, in order to prove the efficiency of our proposed method, we carried out various experiments and explored its efficiency for the tasks of traffic classification, traffic description, and application identification compared to two of the most famous

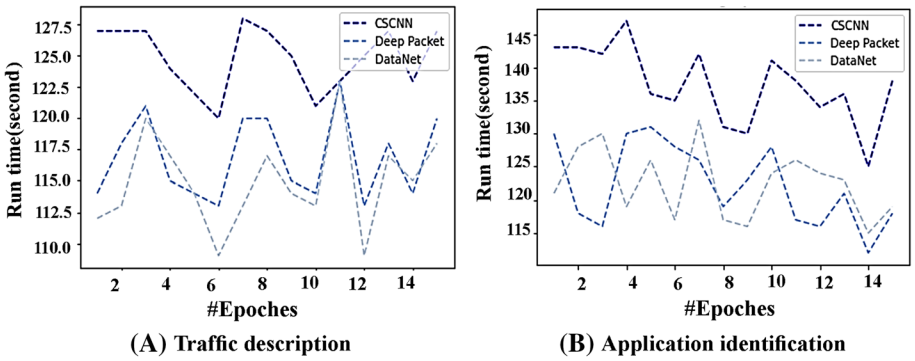

Fig. 10 Training time comparison based on the number of epochs

Table 5 Comparison between CSCNN and other existing methods on ISCX VPN-nonVPN" dataset

Method	Application	Metric	Result
CSCNN	Application Identification	Accuracy	97.9
K-NN [33]			93.94
CSCNN	Traffic description	Precision	97.7
C4.5 [17]			90.0

methods, namely Deep Packet [1] and Datanet [37]. Apparently, CSCNN presented higher performance while its F1-score was about 97.4 and 96.3 % respectively for traffic description and application identification which is not only higher than the two other methods (Tables 3 and 4) but also implies that it is capable of accurately classifying the packets.

In order to provide more comparison, we aimed at comparing our proposed method with other existing methods that evaluated their methods on the "ISCX VPN-nonVPN" dataset. As mentioned in Sect. 2, Gil et al. [17] and Yamansavascilar et al. [33] used this dataset in their experiments. However, it must be emphasized that they utilized handcraft features based on network traffic flow while CSCNN does not require any hand-craft features and consider the network traffic at the packet level and therefore can be more applicable in real-world applications. Table 5 includes the results for comparing the performance of CSCNN with other existing methods. Having the previously mentioned analysis in our mind, it can be concluded that CSCNN has higher performance than both machine learning and deep learning based methods.

It is worth mentioning that Wang et al. [39] also proposed a similar method for traffic description on ISCX VPN-nonVPN" dataset and obtained 100 % precision. However, their obtained result is seriously questionable because their best result was obtained by utilizing packets having all headers from every five layers of the Internet protocol stacks. Considering the fact that the source and destination IP addresses are unique for each application, they presumably only utilized this feature for classification and in that case a much simpler classifier could properly handle the classification task. Particularly, we masked IP address fields in our pre-processing steps to avoid this phenomenon.

6 Conclusion

By the rapid development of the Internet and particularly online applications, accurately classifying Internet traffic has changed to one of the prominent issuers in the field of the computer network. On the other hand, by the enormous growth of deep learning models in various application and their remarkable results considering the fact that they do not need any handcraft features and are able to learn a high representation of input data besides extracting precious features automatically, they have also obtained considerable attention for the task of traffic classification.

Due to the fact that the standard learning methods are particularly designed to minimize the overall error without considering the class distribution, they are generally biased toward the majority classes and result in less sensitivity to minority class samples. In this regard, convergence and generalization of a classification method can be easily influenced by the problem of unbalanced data. To fill this lacuna, a Cost-Sensitive Convolutional Neural Network (CSCNN) is proposed in this paper that tries to deal with the class imbalance issue

in encrypted traffic classification. Accordingly, a cost matrix is generated based on the class distribution which is then utilized during the training process to modify the weights. In order to prove the efficiency of our proposed method, it was compared with machine learning and deep learning based methods on ISCX VPN-nonVPN" dataset for the task of traffic classification, traffic description, and application identification. Based on the results of experiments, it can be concluded that CSCNN has higher efficiency compared to both machine learning and deep learning based methods.

Following a similar line of research, CSCNN can be utilized in various complex operations including multi-channel classification, distinguishing between different types of Skype traffic like chat, voice, and video calls, and Tor traffic. Performing a cost-sensitive strategy to other models like SAE or RNN can be also worth exploring.

References

1. Lotfollahi M, Siavoshani MJ, Zade RSH, Saberian M (2020) Deep packet: A novel approach for encrypted traffic classification using deep learning. Soft Comput 24(3):1999–2012
2. Wang P, Chen X, Ye F, Sun Z (2019) A survey of techniques for mobile service encrypted traffic classification using deep learning. IEEE Access 7:54024–54033
3. D'Angelo G, Palmieri F (2021) Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction. J Netw Comput Appl 173:102890
4. Aceto G, Ciuonzo D, Montieri A, Pescapé A (2021) DISTILLER: Encrypted traffic classification via multimodal multitask deep learning. J Netw Comput Appl:102985
5. Dias KL, Pongelupe MA, Caminhas WM, de Errico L (2019) An innovative approach for real-time network traffic classification. Comput Netw 158:143–157
6. Soleymanpour S, Sadr H, Beheshti H An Efficient Deep Learning Method for Encrypted Traffic Classification on the Web. In: 2020 6th International Conference on Web Research (ICWR) (2020) IEEE, pp 209–216
7. Sadr H, Nazari Solimandarabi M, Mirhosseini Moghadam M (2017) Categorization of persian detached handwritten letters using intelligent combinations of classifiers. J Adv Comput Res 8(4):13–21
8. Sadr H, Pedram MM, Teshnehlab M (2021) Convolutional neural network equipped with attention mechanism and transfer learning for enhancing performance of sentiment analysis. J AI Data Mining. https://doi.org/10.22044/jadm.2021.9618.2100
9. Sadr H, Soleimandarabi MN, Pedram M, Teshnelab M Unified Topic-Based Semantic Models: A Study in Computing the Semantic Relatedness of Geographic Terms. In: 2019 5th International Conference on Web Research (ICWR) (2019) IEEE, pp 134–140
10. Höchst J, Baumgärtner L, Hollick M, Freisleben B Unsupervised traffic flow classification using a neural autoencoder. In (2017) IEEE 42nd Conference on Local Computer Networks (LCN), 2017. IEEE, pp 523–526
11. Bi Q, Zhang H, Qin K (2021) Multi-scale stacking attention pooling for remote sensing scene classification. Neurocomput 436:147–161
12. Wang Q, Huang W, Xiong Z, Li X (2020) Looking Closer at the Scene: Multiscale Representation Learning for Remote Sensing Image Scene Classification. IEEE Transactions on Neural Networks and Learning Systems
13. Jadidinejad AH, Sadr H (2015) Improving weak queries using local cluster analysis as a preliminary framework. Indian J Sci Technol 8(5):495–510
14. Sadr H, Nazari Solimandarabi M (2019) Presentation of an efficient automatic short answer grading model based on combination of pseudo relevance feedback and semantic relatedness measures. J Adv Comput Res 10(2):1–10
15. Sadr H, Pedram MM, Teshnehlab M (2019) A Robust Sentiment Analysis Method Based on Sequential Combination of Convolutional and Recursive Neural Networks. Neural Process Lett:1–17
16. Wang Q, Liu S, Chanussot J, Li X (2018) Scene classification with recurrent attention of VHR remote sensing images. IEEE Trans Geosci Remote Sens 57(2):1155–1167

17. Draper-Gil G, Lashkari AH, Mamun MSI, Ghorbani AA Characterization of encrypted and vpn traffic using time-related. In: Proceedings of the 2nd international conference on information systems security and privacy (ICISSP) (2016) pp 407–414
18. D'Alconzo A, Drago I, Morichetta A, Mellia M, Casas P (2019) A survey on big data for network traffic monitoring and analysis. IEEE Trans Netw Serv Manage 16(3):800–813
19. Qi Y, Xu L, Yang B, Xue Y, Li J Packet classification algorithms: From theory to practice. In: IEEE INFOCOM 2009, 2009. IEEE, pp 648–656
20. Dainotti A, Pescape A, Claffy KC (2012) Issues and future directions in traffic classification. IEEE Network 26(1):35–40
21. Madhukar A, Williamson C A longitudinal study of P2P traffic classification. In: 14th IEEE International Symposium on Modeling, Analysis, and Simulation (2006) IEEE, pp 179–188
22. Moore AW, Papagiannaki K Toward the accurate identification of network applications. In: International Workshop on Passive and Active Network Measurement (2005) Springer, pp 41–54
23. Sherry J, Lan C, Popa RA, Ratnasamy S, Blindbox: Deep packet inspection over encrypted traffic. In: Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, 2015. pp 213–226
24. Hua N, Song H, Lakshman T Variable-stride multi-pattern matching for scalable deep packet inspection. In: IEEE INFOCOM 2009, 2009. IEEE, pp 415–423
25. Wang X, Jiang J, Tang Y, Liu B, Wang X, StriD²FA: Scalable Regular Expression Matching for Deep Packet Inspection. In: 2011 IEEE International Conference on Communications (ICC) (2011) IEEE, pp 1–5
26. Soleimandarabi MN, Mirroshandel SA (2015) A novel approach for computing semantic relatedness of geographic terms. Indian J Sci Technol 8(27):1–11
27. Piskac P, Novotny J Using of time characteristics in data flow for traffic classification. In: IFIP International Conference on Autonomous Infrastructure, Management and Security (2011) Springer, pp 173–176
28. Yildirim T, Radcliffe P VoIP traffic classification in IPSec tunnels. In: 2010 International Conference on Electronics and Information Engineering, 2010. IEEE, pp V1-151-V151-157
29. Crotti M, Dusi M, Gringoli F, Salgarelli L (2007) Traffic classification through simple statistical fingerprinting. ACM SIGCOMM Comput Commun Rev 37(1):5–16
30. Wang X, Parish DJ Optimised multi-stage tcp traffic classifier based on packet size distributions. In: 2010 Third International Conference on Communication Theory, Reliability, and Quality of Service, 2010. IEEE, pp 98–103
31. Auld T, Moore AW, Gull SF (2007) Bayesian neural networks for internet traffic classification. IEEE Trans Neural Netw 18(1):223–239
32. Sun R, Yang B, Peng L, Chen Z, Zhang L, Jing S Traffic classification using probabilistic neural networks. In: 2010 Sixth International Conference on Natural Computation, 2010. IEEE, pp 1914–1919
33. Yamansavascilar B, Guvensan MA, Yavuz AG, Karsligil ME Application identification via network traffic classification. In: 2017 International Conference on Computing, Networking and Communications (ICNC) (2017) IEEE, pp 843–848
34. Chen Z, He K, Li J, Geng Y Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks. In (2017) IEEE International Conference on Big Data (Big Data), 2017. IEEE, pp 1271–1276
35. Wang W, Sheng Y, Wang J, Zeng X, Ye X, Huang Y, Zhu M (2017) HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. IEEE Access 6:1792–1806
36. Wang Q, Wan J, Yuan Y (2017) Deep metric learning for crowdedness regression. IEEE Trans Circuits Syst Video Technol 28(10):2633–2643
37. Wang P, Ye F, Chen X, Qian Y (2018) Datanet: Deep learning based encrypted network traffic classification in sdn home gateway. IEEE Access 6:55380–55391
38. 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
39. Wang W, Zhu M, Wang J, Zeng X, Yang Z End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In (2017) IEEE International Conference on Intelligence and Security Informatics (ISI), 2017. IEEE, pp 43–48
40. Krawczyk B, Woźniak M, Schaefer G (2014) Cost-sensitive decision tree ensembles for effective imbalanced classification. Appl Soft Comput 14:554–562
41. Chung Y-A, Lin H-T, Yang S-W (2015) Cost-aware pre-training for multiclass cost-sensitive deep learning. arXiv preprint arXiv:151109337

42. Buda M, Maki A, Mazurowski MA (2018) A systematic study of the class imbalance problem in convolutional neural networks. Neural Netw 106:249–259
43. Wang S, Liu W, Wu J, Cao L, Meng Q, Kennedy PJ Training deep neural networks on imbalanced data sets. In (2016) international joint conference on neural networks (IJCNN), 2016. IEEE, pp 4368–4374
44. Khan SH, Hayat M, Bennamoun M, Sohel FA, Togneri R (2017) Cost-sensitive learning of deep feature representations from imbalanced data. IEEE Trans Neural Netw Learn syst 29(8):3573–3587
45. Telikani A, Gandomi AH (2019) Cost-sensitive stacked auto-encoders for intrusion detection in the Internet of Things. Internet of Things:100122
46. Sadr H, Solimandarabi MN, Pedram MM, Teshnehlab M (2021) A Novel Deep Learning Method for Textual Sentiment Analysis. arXiv preprint arXiv:210211651
47. Wang Q, Wan J, Yuan Y (2018) Locality constraint distance metric learning for traffic congestion detection. Pattern Recogn 75:272–281
48. Sadr H, Pedram MM, Teshnehlab M (2020) Multi-View Deep Network: A Deep Model Based on Learning Features From Heterogeneous Neural Networks for Sentiment Analysis. IEEE Access 8:86984–86997
49. Sadr H, Pedram MM, Teshnelab M (2019) Improving the performance of text sentiment analysis using deep convolutional neural Network Integrated with Hierarchical attention layer. Int J Inf Commun Technol Res 11(3):57–67