MISCNN : A Novel Learning Scheme for CNN-Based Network Traffic Classification

Ui-Jun Baek
Computer Information and Science
Korea University
Sejong, Korea
pb1069@korea.ac.kr

Boseon Kim
Computer Information and Science
Korea University
Sejong, Korea
boseon12@korea.ac.kr

Jee-Tae Park
Computer Information and Science
Korea University
Sejong, Korea
pjj5846@korea.ac.kr

Jeong-Woo Choi
Computer Information and Science
Korea University
Sejong, Korea
choigoya97@korea.ac.kr

Myung-Sup Kim
Computer Information and Science
Korea University
Sejong, Korea
tmskim@korea.ac.kr

***Abstract*— By the rapid development of the Internet and online applications, traffic classification has changed to an important topic in the field of network management. Although many studies have been conducted in recent years, designing a robust classification model remains a major challenge. Even though previous researches have focused on changing the layer structure within the deep learning model, they do not consider the input shape that best represents the traffic. To this end, a new traffic classification method is presented in this paper that aims to utilize various input shape that can be derived from fixed-length packet bytes. The proposed method utilized MISCNN (Multi Input Shape Convolution Neural Network) to generate robust traffic classification model that can be used in many domains. Various experiments were carried out to verify superiority of proposed method for the tasks of traffic classification and application identification. According to the obtained results, MISCNN achieved higher score compared to previous researches that utilized only 2D square input shape and 1D linear input shape on the ISCX VPN-nonVPN dataset.**

Keywords— traffic classification, traffic identification, convolutional neural network, DL-based classification

I. INTRODUCTION

Traffic Classification (TC) is generally known as a core technology in network traffic monitoring and analysis (NTMA). TC is a technology that aims at grouping similar or related traffic and classifying it into predefined categories, such as normal or malicious traffic, application types, or application protocols or services to which the application belongs [1]. TC is important because of varied reasons that involve:

1) Troubleshooting : locating faulty network devices and hardware/software misconfigurations and point of packet losses, network errors, etc.

2) Quality of Service(QoS) : managing to guarantee the overall acceptability of an application (bandwidth resources, cloud service usage, etc.).

3) Security : distinguishing malicious and abnormal network traffic for network security measurement and intrusion detection.

TC methods that have been widely used since the past include port-based classification and payload-based classification [2]. However, both two types of the methods have clear limitations. Payload-based method cannot classify encrypted traffic, but most of the traffic is encrypted and sent in present. Port-based methods can only classify traffic based on known ports, but many traffic hides ports or uses dynamic ports. In order to overcome the former limitations, Machine Learning (ML) emerged as a suitable alternative for the traffic classification task. Although ML based methods solve the fundamental limitations of former researches, they are still confronted with crucial challenges. Meaningless features from the data prevent classification models from learning and reasoning accurately, so they necessarily require proper feature engineering. Also, ML based models are more prone to overfitting given unbalanced data and cannot accurately learn traffic data from modern dynamic network environments. Furthermore, the number of internet users and applications intensifies the aforementioned challenges [3]. In recent years, deep learning (DL) based methods have get attention for TC owing to the fact that they do not need handcraft features and achieve high classification performance. Among DL methods, convolutional neural network (CNN), which extracts features by converting fixed-length bytes into images, is widely used in recent studies. Most of the studies reshape the raw packet bytes into a square 2D vector, but overlook the fact that the packet structure does not have 2D spatial information unlike a typical image used in computer vision domain. Some studies report that 1D-based CNN outperform 2D-based CNN, but there are not enough considerations for input shapes [10].

This paper assumes that there will be other input shapes that best represent the characteristics of the packet, and proposes MISCNN, which simultaneously learns various input shapes that can be derived from a fixed length of bytes. Based on the empirical results, it can be claimed that MISCNN has superior classification performance than existing model that adopts square input shape or 1D input shape.

The remainder of the paper consists of related studies, datasets used in experiments, descriptions of the proposed model, experiments and results and conclusion.

This results was supported by "Regional Innovation Strategy (RIS)" through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(MOE)(2021RIS-004) and was supported by Electronics and Telecommunications Research Institute(ETRI) grant funded by the Korean government (No.22RH1210, Development of Cloud-based Autonomic Federation Service Platform Architecture/Core Technology Design/Validation in the 5G & Beyond Era)

TABLE I. CNN-BASED TC STUDIES

Paper No.	Category	DL Method	Features	Input shape
[7] (2018)	IDS	CNN	Header, Payload	S
[8] (2018)	APP TC	CNN	Header	S
[9] (2019)	APP TC	CNN	Payload	S
[10] (2019)	APP TC	CNN	Payload	S, L
[11] (2020)	APP TC	CNN, SAE	Header	L
[12] (2020)	APP TC	CNN	Header, Payload	S
[13] (2021)	APP TC	CNN	Header	L
[14] (2021)	APP TC	CNN, LSTM	Header	L
[15] (2021)	APP TC	CNN, LSTM	Payload	S, L
[16] (2022)	APP TC	CNN	Header (Statistic)	L

※ S: Square like, L: Linear

II. RELATED WORK

The deep neural network (DNN) is an artificial neural network (ANN) consisting of multiple hidden layers between the input layers and output layers [4]. CNN and RNN are mainly utilized in the TC field [5], but this section only briefly covers the description of CNN.

CNN is a DL method that supplements the problems that occur when processing data such as images or videos in general DNNs. CNN mainly consists of three layers, as follows:

1) Convolution layer: Extracting local features from image using filter.

2) Pooling layer: Memorizing only representative values (Max, Avg, etc.) with in a pre-fixed window in order not to be affected by changes in the topology of the image.

3) Fully-connected layer: A layer that learns global features created through iteration of local feature extraction of convolutional layers and pooling layers.

CNN can learn spatial features and have already achieved impressive results in many computer vision tasks, such as image classification [6].

CNN is the most commonly used method for DL based traffic classification, and packets are truncated by fixed length to enter the learning model and are usually converted to 2D vector of square-shape and 1D vector of linear-shape. There are former studies to classify network traffic using CNN, which are summarized in Table 1. According to our investigations, in the past two-dimensional square-like input shapes were mainly used. Relatively recently, 1-D linear input shapes have been constantly applied, presumably based on empirical studies that show that linear input shape-based TC model perform better than square-like input shape-based [10]. As far as we know, there are no studies using input shapes that are different from the two input shape type mentioned above.

TABLE II. TC TASKS OF ISCX VPN-NONVPN

Task	Classes
Encapsulation	nonVPN, VPN
Traffic Type	VoIP, FileTransfer, P2P, Streaming, Chat, Email, Browser
Application	Skype, Torrent, Hangouts, VoipBuster, Facebook, FTPs, SCP, Email, Youtube, Vimeo, Spotify, Netflix, SFTP, Aim, ICQ

TABLE III. ISCX VPN-NONVPN 2016 DATASET

Category	Application	Encapsulation		Category ratio
		NonVPN	VPN	
Chat	aim chat	421	32	14,396 (4.68%)
	facecbook chat	509	1,159	
	hangout chat	438	2,751	
	icq chat	438	31	
	skype chat	8,561	56	
Email	email	7,314	298	8,061 (2.62%)
	gmail	449	-	
Streaming	netflix	564	173	2,761 (0.89%)
	vimeo	433	136	
	youtube	879	213	
	spotify	226	137	
File transfer	skype file	56,930	867	67,353 (21.9%)
	scp	8355	-	
	sftp	218	28	
	ftp	830	125	
VoIP	skype audio	38533	912	214,045 (69.6%)
	skype video	571	-	
	hangout audio	78,996	8,120	
	hangout video	1,513	-	
	facebook audio	79,488	1,355	
	Voi p buster	2,938	1,619	
Browser	facebook video	414	-	414 (0.13%)
P2P	bit torrent	-	477	477 (0.16%)
VPN-nonVPN Ratio		289,018 (93.98%)	18,489 (6.01%)	

III. MISCNN

A. Dataset

The "ISCX VPN-nonVPN 2016" dataset with raw pcap format of various applications is used to evaluate the classification performance [17]. It includes human-generated traffic encompassing different traffic types with information also on the related applications collected through both regular sessions and sessions encapsulated over VPN. In view of this structure, we can associate a three-view label (i.e. encapsulation, traffic type, and application) to whatever segmentation of raw network traffic (i.e. to a generic TC object). Such three-view label corresponds to just as many TC tasks to be tackled. Table 2 lists the ISCX VPN-nonVPN classes associated to each task. Each task is applied in the proposed model and comparative experiments.

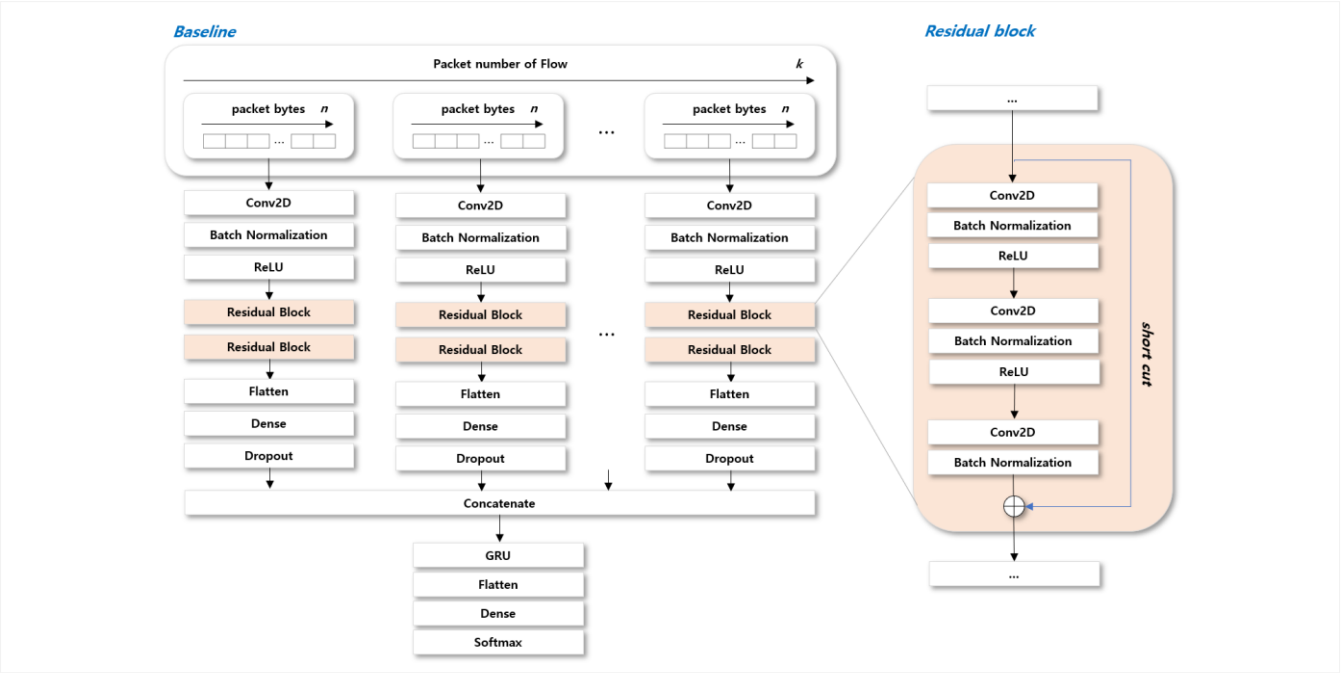

Fig. 1. Baseline

B. *Preprocess*

Packets extracted from the raw pcap file are aggregated into flows based on the 5-tuple(source IP, destination IP, source port, destination port, protocol) and reconstructed into bidirectional flows with considered directionality. Nearly 60 percent of the aggregated two-way flows contain only one UDP packet, which interfere with proper learning process and have been eliminated [15]. As a result, 27.8k bidirectional flows were extracted and the distribution is shown in Table 3. According to Table 3, the dataset is highly imbalanced, which can adversely affect learning process, so under-sampling or over-sampling can be considered [18]. However, imbalance occurs frequently in real network environments, so we did not perform any sampling process.

Each flow sample contains only the first-*k* packets among all packets, and when the number of packets in the flow is less than pre-fixed *k*, an empty object filled with zero is added. The First-*n* bytes are extracted from the packets contained in the flow, and when the size of packet is less than pre-fixed n, the remaining empty space of packet object is padded to zero. The initial shape of packet data is a *k**1, which is reshaped as *p***q* in the Reshape layer.

C. *Baseline*

The baseline used in this study is a mixed model of CNN and Gated Recurrent Unit (GRU), and is shown in Figure 1. Using each packet as an input rather than aggregating it as a single input ensures high performance [7]. Therefore, the baseline receives multiple packets and then merges into one output at the GRU layer. Residual block was placed to extract features from the input made up of each packet [19]. Each residual block consists of three Convolution Layer and after each convolution operation, Batch Normalization layer is placed to adjust the mean and variance of the output. The hyper parameters of the baseline are show in Table 4.

TABLE IV. HYPERPARAMETERS OF BASELINE

Layer	Hyperparameters
Conv2D	Kernel size: (8,1), n_filters:4
Residual Block[a]	-
Dense	Unit:32
Dropout	Rate:0.2
GRU	Unit:16
Dense	Units:6

a. The hyperparameters of the residual block depend on the model and are given in Table 5.

Fig. 2. MISCNN Scheme

TABLE VII. COMPARISON OF MISCNN WITH PREVIOUS STUDEIS

	Encapsulation (%)		Traffic Type (%)		Application (%)		RTPE[s]
	Accuracy	F1 score	Accuracy	F1 score	Accuracy	F1 score	
1D-CNN [20]	87.47	83.50	73.14	71.14	72.73	61.35	13.83
1D-CNN [22]	82.39	76.24	56.09	54.75	56.54	40.87	1.70
2D-CNN [21]	87.43	83.51	71.86	69.77	71.45	59.29	40.9
Distiller [15]	93.75	91.95	80.78	78.72	77.63	66.44	5.99
Baseline(1D)	97.47	95.28	91.21	91.36	81.08	78.79	23.5
Baseline(2D)	94.12	94.04	87.22	85.23	78.06	77.65	23.5
MISCNN	98.58	98.58	93.58	93.21	85.02	85.26	40.1
MISCNN Gain	+4.83	+6.63	+12.8	+14.49	+7.39	+18.82	+34.11

[a.] RTPE: Run-Time Per Epoch

TABLE V. HYPERPARAMETERS OF MISCNN

Shape	kernel_size	n_filters
(784, 1)	(6, 1)	Initial value : 4, Last value : 16 The initial value is 4, Which doubles for each residual block.
(392, 2)	(6, 1)	
(196, 4)	(4, 1)	
(98, 8)	(4, 1)	
(56, 14)	(2, 1)	
(49, 16)	(2, 2)	
(28, 28)	(2, 2)	
(16, 49)	(2, 2)	
(14, 56)	(1, 4)	
(8, 98)	(1, 4)	
(7, 112)	(1, 8)	
(4, 196)	(1, 10)	
(2, 392)	(1, 20)	

D. MISCNN

MISCNN is based on a very simple idea and it is powerful. The input before it is passed to the first residual block is split into inputs with multiple shapes. The number of shapes each input can have is equal to the number of divisor in the packet vector size *n*. For example, 784*1 input that are frequently used in previous studies have 15 different shapes (784*1, 392*2, 196*4, … , 1*784). At this point, shape 1*784 is excluded because it has the same structure as shape 784*1 (but shape 2*392 and shape 392*2 are clearly different). An overview of the MISCNN is shown in Figure 2. These reshape methods are not available in the field of image recognition and computer vision because they compromise the 2D spatial information of adjacent(vertical, horizontal) pixels of a typical image. On the other hand, because raw packet do not have 2D spatial information, reshape methods can be considered, which allows the training model to observe the same input from various perspectives. The hyperparameters vary for each model, as shown in Table 5. Each shape is used as an input to the residual block, and the shape's hyperparameters correspond to the hyperparameters of each residual block. When *n* is 784, The hyperparameters of each shape are shown in Table 5.

E. Experiment setup

The hardware and software of the environment in which the training process was performed are shown in Table 6. When compiling the model, the Learning Rate was set to 25*e*-5. The loss function was set to *Categorial Crossentropy*, and *Adam Optimizer* was used as the optimizer. The classification performance was measured according to the change in the number of packet bytes *n* and the number of packets *k*, and *p* and *q* are dependent on *n*. To address extreme class imbalances, we set *class weight* corresponding to the number of sample per class.

TABLE VI. EXPERIMENT ENVIRONMENT

	List	Spec
Hardware	CPU	Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz
	GPU	NVIDIA GP102 [TITAN Xp]
	RAM	120G
Software	Nvidia driver	440.33.01
	CuDNN	cuDNN/7.6 for cuda 10.1
	Cuda	cuda/10.1
	Python	python3.6.9
	Tensorflow	tensorflow 2.3.0
	Keras	keras 2.4.0

IV. EXPERIMENTAL RESULTS

A. Metrics

Accuracy, *Precision*, *Recall* and *F-measure* are used for evaluation, and are follows:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$Precision = \frac{TP}{TP+FP} \quad (2)$$

$$Recall = \frac{TP}{TP+FN} \quad (3)$$

$$F1\ score = 2 * \frac{Recall*Precision}{Recall+Precision} \quad (4)$$

B. Overall comparison with previous studies

To verify the effectiveness of the proposed method, MISCNN and four previous studies and baselines were compared by three TC tasks, and comparison results are shown in Table 7. It is noteworthy that MISCNN outperformed previous studies. Compared to the best existing methods in the three tasks, it showed a rise of 4.83% in *Encapsulation* task, more than 10% in *Category* task, and 7% in *Application* task, which is the most difficult task. In particular, F-measure increased significantly compared to previous study. Another thing to note is that Baseline's performance also outperformed previous TC models. Baseline applied with skip-connection method, which showed excellent performance in the field of image recognition and computer vision, shows better performance than previous TC models in two metrics, which proves that skip-connection method is useful in the field of TC. One last observation is that the 1D CNN-based TC model extracts traffic characteristics better than the 2D CNN-based TC model, which is the same as the results of previous studies.

TABLE VIII. COMPARISON BY EXPERIMENTAL PARAMETERS (ENCAPSULATION)

	n=484 (*shp=8*)				n=576 (*shp=20*)				n=676 (*shp=8*)				n=784 (*shp=14*)			
k	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**
3	97.4	97.4	97.4	97.4	98.2	98.2	98.2	98.2	97.7	97.7	97.7	97.7	98.0	98.0	98.0	98.0
5	98.3	98.3	98.3	98.3	98.0	98.0	98.0	98.0	97.8	97.8	97.8	97.8	98.3	98.3	98.3	98.3
7	98.3	98.3	98.3	98.4	98.4	98.2	98.2	98.2	98.2	98.2	98.2	98.2	98.2	98.2	98.2	98.2
10	**98.6**	**98.5**	**98.6**	**98.6**	98.4	98.3	98.3	98.3	**98.6**	**98.6**	**98.6**	**98.6**	98.3	98.3	98.3	98.3

Acc: Accuracy, Prc: Precision, Rc: Recall, Fm: F-measure

※ Shp: Number of models derived from one input

TABLE IX. COMPARISON BY EXPERIMENTAL PARAMETERS (CATEGORY)

	n=484 (*shp=8*)				n=576 (*shp=20*)				n=676 (*shp=8*)				n=784 (*shp=14*)			
k	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**Fm**	**Acc**	**Prc**	**Rc**	**F1**
3	91.3	91.6	91.3	91.4	90.5	90.6	90.4	90.5	88.1	88.5	88.0	88.2	86.9	87.0	86.8	86.9
5	89.8	90.0	89.8	89.9	91.1	91.3	91.1	91.2	8801	88.3	88.0	88.9	91.7	91.9	91.7	91.8
7	92.0	92.1	91.8	92.0	**92.1**	**92.3**	**92.1**	**92.2**	88.8	89.1	88.7	88.9	92.0	91.9	91.7	91.8
10	90.4	90.6	90.3	90.4	92.1	92.2	92.1	92.1	87.4	87.6	87.3	87.5	88.5	88.7	88.4	88.5

TABLE X. COMPARISON BY EXPERIMENTAL PARAMETERS (APPLICATION)

	n=484 (*s=8*)				n=576 (*shp=20*)				n=676 (*shp=8*)				n=784 (*shp=14*)			
k	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**	**Acc**	**Prc**	**Rc**	**F1**
3	82.4	84.2	80.1	82.5	84.6	86.0	83.7	84.8	80.4	82.6	78.2	80.3	81.8	84.0	80.2	82.1
5	82.6	84.0	81.4	82.7	84.4	85.6	83.7	84.6	84.6	86.0	83.4	84.7	83.9	85.5	82.0	83.7
7	83.0	84.8	82.5	83.6	**85.0**	**86.2**	**84.4**	**85.3**	82.9	84.6	81.4	82.9	82.5	84.6	81.2	82.9
10	83.1	84.7	81.6	83.1	83.9	85.1	83.2	84.1	83.5	84.9	82.5	83.7	84.2	85.3	83.3	84.3

C. Comparison by experimental parameters

In this section, the TC performance by the *n* (fixed packet length) and *k* (=the number of packets input to the TC model) and is summarized for each of the three tasks. *shp* means the number of different shapes derived from one input.

First, in the encapsulation task, there was no significant performance change by *n*, and the performance slightly increased as *k* increased. The best performance is when *n* is 484 or 676, where the number of *shp* is equal to 8. This suggests that too many shapes cause overfitting of the TC model, and it is not recommended to use too many shapes in *Encapsulation* task, which is a relatively easy task. In the *Category* task, no performance changes are observed as *n* increases or decreases, significant change is observed depending on the *shp*. When *n* is 576 and *n* is 784, the *shp* is 20 and 14, respectively, showing the best performance. In addtion, as k increases, performance increases, and then when k is 7, it shows the best performance, and if it increases further performance deteriorates, rather. In the application task, the TC model shows the highest classification performance when *n* is 576, and no clear pattern was observed with changes in *n, k, shp.* From a global perspective, it can be observed that the higher the number of shapes shows higher TC performance, indicating that the proposed method is useful in TC field.

V. CONCLUSION

In this paper, we propose a DL learning scheme to observe packets from many perspectives through various shapes that can be derived from one input. We focused on packet data not having two dimensional spatial information, unlike typical images, and assumed that reshaping packets into various shapes is a way to observe and extract the useful features of raw packets. We conducted an evaluation using an open dataset ISCX VPN-nonVPN 2016 for verification of the proposed method and designed a baseline that applied skip-connection method. Finally, we implement an MISCNN applying the proposed method to the designed baseline and compare TC performance with previous studies.

Our proposal has three contributions. First, overall comparison show that MISCNN improved 14% in accuracy and 18% in f-measure over state-of-the-art TC methods. This performance improvement highlights that observing packets from more diverse perspectives prevents overfitting of the TC model and leads to higher performance improvements. Second, performance comparisons based on experimental parameter changes show that using many shapes, except for *Encapsulation* task, helps classify accurately. In fact, considering that Encapsulation task are less important and more versatile than other tasks in real world networks, bringing to many shapes into TC model can be useful in common situations. Last, comparative experiments were conducted within an unprecedented dataset of preprocessing for data imbalances, which mimicked real-world networks. This means that the proposed MISCNN is practical enough to be used in real-world network environments.

Three future works are presented in connection with this study. The first is lightening. MISCNN is accurate but excessively heavy, which undermines the practicality of the TC model. Secondly, the application of state-of-the-art. Third, how to handle cases where there are no classes or ground truth.

REFERENCES

[1] J. Zhao, X. Jing, Z. Yan, and W. Pedrycz, "Network traffic classification for data fusion: A survey," Information Fusion, vol. 72, pp. 22–47, Aug. 2021, doi: 10.1016/j.inffus.2021.02.009.

[2] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A Survey of Payload-Based Traffic Classification Approaches," IEEE Communications Surveys & Tutorials, vol. 16, no. 2, pp. 1135–1156, 2014, doi: 10.1109/SURV.2013.100613.00161.

[3] T. Barnett, S. Jain, U. Andra, and T. Khurana, "Cisco visual networking index (vni) complete forecast update, 2017–2022," Americas/EMEAR Cisco Knowledge Network (CKN) Presentation, 2018.

[4] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," IEEE transactions on pattern analysis and machine intelligence, vol. 35, no. 8, pp. 1798–1828, 2013.

[5] S. Rezaei and X. Liu, “Deep Learning for Encrypted Traffic Classification: An Overview,” IEEE Communications Magazine, vol. 57, no. 5, pp. 76–81, May 2019, doi: 10.1109/MCOM.2019.1800819.

[6] J. Gu et al., “Recent advances in convolutional neural networks,” Pattern Recognition, vol. 77, pp. 354–377, 2018.

[7] W. Wang et al., “HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection,” IEEE access, vol. 6, pp. 1792–1806, 2017.

[8] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, “Mobile encrypted traffic classification using deep learning,” in 2018 Network traffic measurement and analysis conference (TMA), 2018, pp. 1–8.

[9] H.-K. Lim, J.-B. Kim, J.-S. Heo, K. Kim, Y.-G. Hong, and Y.-H. Han, “[9] Packet-based network traffic classification using deep learning,” in 2019 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), 2019, pp. 046–051.

[10] L. Xu, X. Zhou, Y. Ren, and Y. Qin, “A traffic classification method based on packet transport layer payload by ensemble learning,” in 2019 IEEE Symposium on Computers and Communications (ISCC), 2019, pp. 1–6.

[11] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, “Deep packet: a novel approach for encrypted traffic classification using deep learning,” Soft Comput, vol. 24, no. 3, pp. 1999–2012, Feb. 2020, doi: 10.1007/s00500-019-04030-2.

[12] X. Hu, C. Gu, and F. Wei, “CLD-Net: a network combining CNN and LSTM for internet encrypted traffic classification,” Security and Communication Networks, vol. 2021, 2021.

[13] F. Sarhangian, R. Kashef, and M. Jaseemuddin, “Efficient traffic classification using hybrid deep learning,” in 2021 IEEE International Systems Conference (SysCon), 2021, pp. 1–8.

[14] J. Li and Z. Pan, “Network traffic classification based on deep learning,” KSII Transactions on Internet and Information Systems (TIIS), vol. 14, no. 11, pp. 4246–4267, 2020.

[15] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, “DISTILLER: Encrypted traffic classification via multimodal multitask deep learning,” Journal of Network and Computer Applications, vol. 183, p. 102985, 2021.

[16] S. Izadi, M. Ahmadi, and R. Nikbazm, “Network traffic classification using convolutional neural network and ant-lion optimization,” Computers and Electrical Engineering, vol. 101, p. 108024, 2022.

[17] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, “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] S. Sharma, A. Gosain, and S. Jain, “A Review of the Oversampling Techniques in Class Imbalance Problem,” in International Conference on Innovative Computing and Communications, 2022, pp. 459–472.

[19] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[20] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, “End-to-end encrypted traffic classification with one-dimensional convolution neural networks,” in 2017 IEEE International Conference on Intelligence and Security Informatics (ISI), Jul. 2017, pp. 43–48. doi: 10.1109/ISI.2017.8004872.

[21] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, “Automatic Multi-task Learning System for Abnormal Network Traffic Detection,” Int. J. Emerg. Technol. Learn., vol. 13, no. 04, p. 4, Mar. 2018, doi: 10.3991/ijet.v13i04.8466.

[22] S. Rezaei and X. Liu, “Multitask Learning for Network Traffic Classification,” in 2020 29th International Conference on Computer Communications and Networks (ICCCN), Aug. 2020, pp. 1–9. doi: 10.1109/ICCCN49398.2020.9209652.