Contents lists available at Science-Gate

International Journal of Advanced and Applied Sciences

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

Improved network traffic classification using hashing techniques in machine and deep learning

Mohammed Altaimimi *

Department of Information and Computer Science, College of Computer Science and Engineering, University of Ha'il, Ha'il, Saudi Arabia

ARTICLE INFO

Article history:
Received 31 December 2024
Received in revised form
8 May 2025
Accepted 20 May 2025

Keywords:
Network traffic
Machine learning
Deep learning
Hashing techniques
Traffic classification

ABSTRACT

The rapid global growth of the internet, driven by advancements in fiber and 5G technology, multi-device access, and affordable services, has increased the pressure on internet service providers to classify network traffic efficiently. Accurate traffic classification and protocol identification are critical for detecting malicious activity. This study introduces a new method that enhances machine learning and deep learning models by applying hashing techniques to convert string-based IP addresses into numerical values. The improved models demonstrate a significant boost in accuracy, increasing from 76% to 83%, along with better recall and F1-scores in key categories. These findings highlight the potential of hashing techniques to improve the performance of machine learning models in network traffic classification tasks.

1. Introduction

In recent years, global usage of the internet has grown substantially, with the internet becoming a vital component of daily operations for both individuals and businesses. This surge is driven by advancements in internet technology, such as fiber and 5G connections, easy accessibility across multiple devices, competitive pricing, and a wide range of services available online (Oloyede et al., 2023). As a result, pressure is on internet service providers to develop solutions for classifying network traffic and identifying underlying applications and protocol responsibility, which helps identify malicious activities, malware communication, and unauthorized access attempts. For example, certain malware utilizes specific protocols for communication; therefore, detecting unusual traffic over protocols such as HTTP, FTP, or DNS can assist in identifying malware activities and enhance cybersecurity measures (Nadler et al., 2019).

Machine learning algorithms such as Naive Bayes (NB), as well as deep learning models like convolutional neural networks (CNN) and recurrent neural networks (RNN), often face difficulties handling categorical data, especially when dealing with string-based representations such as IP addresses. The complexity of and potential similarities among these strings can create challenges for both model training and classification accuracy. Regular representation (Guerra et al., 2022) can fail due to correlations between IP addresses (and decisions). This failure may arise from interactions between features, leading to misclassification caused by overlaps between classes.

To overcome these challenges, we propose hashing techniques that can be used to convert categorical string data (IP addresses) into a more manageable numerical form. This transformation both simplifies numerical processing for machine learning models and improves the efficiency and effectiveness of classification tasks. The contributions of this paper can be summarized as follows:

- This study presents a novel method for enhancing NB classification accuracy using hashing techniques.
- The novel method improves communication efficiency by employing numerical compression, which requires fewer resources and less processing than traditional encoding methods.
- The study also provides reviews of both machine learning with statistical features and deep learning

* Corresponding Author.
Email Address: mh.altamimi@uoh.edu.sa
https://doi.org/10.21833/ijaas.2025.05.024
Corresponding author's ORCID profile:
https://orcid.org/0000-0002-4170-6910

approaches, as these are regarded as current trends in network traffic analysis.

The rest of the paper is organized as follows: Section 2 provides an overview of recent studies in the literature and highlights the algorithms and techniques used, and Section 3 introduces the proposed methodology in this study. Section 4 describes the experimental setting used to test the method, and Section 5 discusses the results. Section 6 discusses the results and compares the study outcomes with the literature. Finally, conclusions and suggestions for future study directions are presented in Section 7.

2. Related work

The initial network analysis attempted by Moore and Zuev (2005) utilized an NB model to classify various applications, such as WWW, MAIL, and BULK, achieving a 65.26% accuracy rate. Subsequent efforts were made to enhance the results by incorporating kernel density estimation and fast correlation-based filtering (FCBF), resulting in accuracy rates of 93.50% and 94.29% for the datasets analyzed.

A study conducted by Jenefa and Moses (2018) examined the use of different machine learning algorithms, including C4.5, NB, support vector machine (SVM), and Random Forest (RF), for classifying network flows as Skype or non-Skype. Various features were considered, such as the number of packets and bytes in each direction, packet length, and inter-arrival time. The results showed enhanced performance when utilizing CFS feature selection, with C4.5 showing superiority.

Furthermore, Sun et al. (2018) discussed the limitations of SVMs, such as high training complexity and computational costs. To mitigate these issues, they proposed an incremental SVM (ISVM) as a solution to reduce memory and CPU training costs. A comparison of the performance of their ISVM with other machine learning algorithms, such as NB, demonstrated that it achieves greater accuracy (Sun et al., 2018).

Subsequent research (Cao et al., 2020) proposed an enhanced network traffic classification model based on an SVM. The model tackled the issue of the false removal of combined features caused by traditional feature selection methods and was demonstrated to have high classification accuracy, outperforming NB and k-nearest neighbors (kNN) algorithms. Similarly, a study by de Menezes and de Mello (2021) used statistical and machine learning approaches to classify network flows into 12 classes (defined by Moore and Papagiannaki (2005)) using a kNN classifier and J4.8 decision tree and utilizing features such as the duration, byte count and packet count to identify different traffic types.

Recently, several studies have used deep learning for network classification. In these, network traffic is used as input to a deep learning model to classify protocols within traffic (Azab et al., 2024). One advantage of deep learning over machine learning is that it reduces the need for a feature-engineering phase, which makes it a more convenient and straightforward approach. Wang (2015) was among the early pioneers applying deep learning algorithms for network traffic classification. His framework employed an artificial neural network and a stacked autoencoder (SAE) to classify network traffic based on application types. The results of Wang's (2015) evaluation demonstrated the effectiveness of the proposed approach and showed the potential of further study into the use of deep learning techniques.

Lotfollahi et al. (2020) presented a deep learning framework that leveraged both CNNs and SAEs to classify network traffic. The SAE model consisted of five densely connected layers, while the CNN model employed one-dimensional convolutional layers. Their experiments focused on classifying 12 protocols, including email, chat, torrent, and file transfer, and demonstrated the high effectiveness of the framework, which showed strong performance in application identification using an ISCX dataset created by Sharafaldin et al. (2018). Wang et al. (2018) introduced a deep learning-based framework for classifying encrypted data packets; their proposed classifier 'DataNets' was developed using three different approaches–multilayer perceptron (MLP), SAE and CNN–and was evaluated using ISCX datasets, showing its high accuracy with minimal computational costs. In addition, Liu et al. (2019) proposed an end-to-end framework, called a flow sequence network, for network traffic classification that uses a recurrent neural network to classify encrypted traffic, incorporating an encoder, decoder, and SoftMax classifier. Evaluation of their framework showed it surpasses other state-of-the-art methods in network traffic classification.

An essential aspect of network security is the continuous monitoring of a computer network for unauthorized or unauthenticated access to prevent malicious activity such as attacks. This commonly involves an intrusion detection system (IDS), which notifies cybersecurity analysts and enables them to take the necessary actions. Machine learning is problematic in terms of intrusion detection, as network traffic cannot be directly used as input for machine learning. Thus, the use of machine learning in intrusion detection requires feature engineering (Maxwell et al., 2019).

Feature hashing is a technique for transforming high-dimensional categorical or text data into low-dimensional numerical data using a hash function. This approach is especially useful for handling sparse, high-cardinality data, such as text or categorical features with many unique values (Chi and Zhu, 2017). In contrast, other feature-engineering methods, such as encoding methods that convert categorical data into numerical values, can be resource-intensive, often requiring substantial memory and computational resources (Hancock and Khoshgoftaar, 2020). While a review of the literature suggests that deep learning outperforms traditional

machine learning approaches, such as NB. One method for improving the performance of NB is to incorporate the encoding of features for new categories such as one-hot encoding, focus on encoding features for new categories. However, with large datasets, this approach may result in the creation of so many categories that performance issues are introduced. In contrast, hashing encodes each unique feature to a fixed-size vector, reducing the need to create multiple categorical variables and thereby optimizing memory usage.

3. Methodology

The NB algorithm is a supervised learning technique grounded in Bayes' theorem of probability. It leverages learning models that create knowledge structures to classify unseen instances into predefined categories, often delivering strong results on small, balanced datasets (Lewis, 1998). However, its classification accuracy can decline with large, imbalanced datasets. Thus, NB performs effectively when there are clear differences in class distributions but may encounter difficulties with certain data types, such as IP addresses.

To mitigate these issues, hashing techniques can be used to convert categorical string data into numerical representations. In the context of machine learning, hashing serves various purposes, including dimensionality reduction, feature encoding, and data privacy assurance. The primary objective of hashing is to transform high-cardinality categorical data (i.e., IP addresses) into numerical representations suitable for machine learning algorithms. This conversion simplifies data handling and improves the efficiency and accuracy of classification tasks, addressing the challenges faced by NB when dealing with complex categorical data. A hash function h is defined as;

$$h: S \to N$$

where,

- S is the set of all possible input strings (e.g., IP addresses)
- $N = \{0, 1, 2, \ldots, B\dashv - 1\}$ is the set of output hash values, with B being the number of buckets.

For a hash function h, we implemented the MD5 hashing algorithm followed by a modulo operation to constrain the hash values within a specified range (number of buckets). Thus,

$$h(IP) = (int(MD5(IP)hexdigest, 16))mod\ B$$

where,

- $MD5(IP)$ computes the MD5 hash of the input IP address string.
- $hexdigest$ converts the MD5 hash to a hexadecimal string.
- $int(.,16)$ interprets the hexadecimal string as a base-16 integer.
- $B = 1{,}000$ is the number of hash buckets.

Consider the IP address '192.168.1.1': The IP is mapped to 777 in the 'Source_hashed' feature as follows (Stallings and Brown, 2015):

1. MD5 hashing:

$$\text{MD5}('192{,}168.1.1.1') = 'c0a80101'$$
(hexadecimal representation)

2. Hexadecimal to integer conversion:

$$int('c0a80101', 16) = 3232235777$$

3. Modulo operation:

$$h('192{,}168.1.1.1' = 3232235777\ mod\ 1000 = 777$$

Thus, '192.168.1.1' is mapped to 777.

4. Experimental setting

This section provides information about the experiments. We begin by explaining the dataset that was used and summarizing the statistics. Following this, we discuss the preprocessing steps taken before training the models. Finally, we present the results and discuss the findings.

4.1. Dataset

The dataset used in the experiments was generated by the University of Cincinnati in Ohio using packet captures on a Kali machine. Stored as a CSV file, a total of 394,137 instances are distributed into seven features: instance number, timestamp, source IP, destination IP, protocol, length, and additional traffic information. While most features are numeric, some are nominal or date-time-based. This dataset can be used for various network-related machine learning applications, such as network traffic classification, network performance monitoring, network security management, and traffic management. The dataset was chosen because it is new and open source. Also, the dataset is well annotated with a few missing labels, which makes it easier to explore and provides a solid foundation for building and refining models. The well-annotated nature of the data ensures more grounded and reliable results.

4.2. Preprocessing

Preprocessing began with extracting features from the temporal data, such as hour, day of week, and time, which were then converted to a datetime format suitable for model classification. Next, the 'length' feature was scaled to normalize the data, which is important for optimal performance of the NB model. Following this, the target variable

'protocol' was label-encoded for conversion into numerical values suitable for machine learning models. We excluded protocol classes with fewer than 1,000 instances to ensure a balanced dataset. Then, five different network protocols were identified for the experiment: DNS, ICMP, TCP, TLSv1.2, and TLSv1.3. After preprocessing the dataset, the features and target variables were separated. The data were then divided into training and testing sets, with 80% allocated for training and 20% allocated for testing. The training set consisted of 314,260 instances, while the testing set consisted of 78,590 instances. The model framework is shown in Fig. 1.

The architecture of the CNN was compiled using a 1D convolutional neural network. To ensure numerical consistency, the IP addresses (source and destination) were hashed using the method described in Section 3. The CNN model consisted of two 1D convolutional layers followed by max-pooling layers, a flatten layer to reduce dimensionality, and a fully connected dense layer. The model was compiled using the Adam optimizer and sparse categorical cross-entropy loss. Similarly, the architecture of the RNN model was used to handle the sequential data. The model's architecture consisted of an RNN layer followed by dense layers with dropout for regularization. The model was compiled using the Adam optimizer and trained with early stopping and a learning rate scheduler to prevent overfitting.

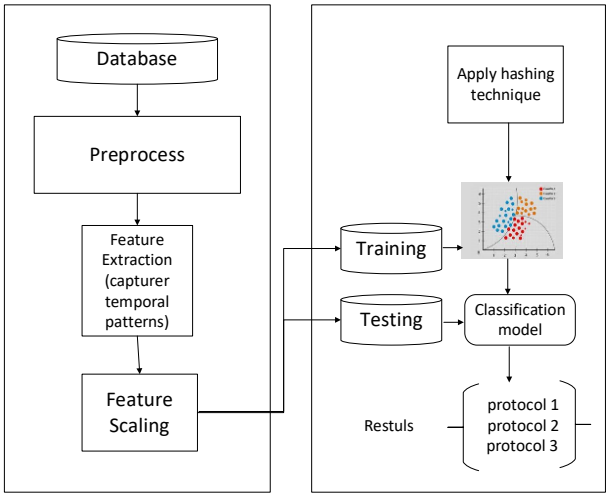

Fig. 1: The improved Naive Bayes-based model for network traffic classification (Cao et al., 2020)

4.3. Evaluation matrix

After training and testing were completed, we employed various metrics to assess the results. Specifically, we computed the confusion matrix for each theme in the experiment. Each element in the matrix represents one of the following: true positives (TP), which are instances where the classification aligned with the correct theme; false positives (FP), in which the classification incorrectly labelled a line as a positive match; true negatives (TN), where the classification correctly identified a negative theme; and false negatives (FN), in which the classification incorrectly labelled a line as a negative match.

Additionally, we evaluated the model's performance using metrics such as accuracy, precision, recall, and F1-score, all based on the confusion matrix (Davis and Goadrich, 2006). Accuracy refers to the proportion of correctly classified lines calculated by dividing the sum of true positives and true negatives by the total number of classified lines:

$$\text{Accuracy} = \frac{(\#\,\text{TP} + \#\,\text{TN})}{(\#\,\text{TP} + \#\,\text{FP} + \#\,\text{FN} + \#\,\text{TN})}.$$

Precision refers to the proportion of selected lines that were accurately identified as positive. It was calculated by dividing the number of true positive lines by the total number of lines classified as positive:

$$\text{Precision} = \frac{(\#\,\text{TP})}{(\#\,\text{TP} + \#\,\text{FP})}.$$

Recall refers to the percentage of actual positive cases that were correctly identified. It was calculated by dividing the number of true positive instances by the total number of actual positive instances:

$$\text{Recall} = \frac{(\#\,\text{TP})}{(\#\,\text{TP} + \#\,\text{FN})}.$$

The F1-score combines both precision and recall into a single metric:

$$\text{F1} = \frac{2 * (\text{Recall} \times \text{Precision})}{(\text{Recall} + \text{Precision})}.$$

5. Results

The classification performances of the two models–NB using encoding teaching and improved NB using hashing–were compared, as were their performances when subjected to 10-fold cross-validation. The improved NB model demonstrated noticeable enhancements in classification performance (Table 1), particularly in terms of overall accuracy and recall. In the first model, using NB encoding, the accuracy was 76%, with a macro average precision of 0.38 and a recall of 0.49. It performed poorly on identifying Class 1, with a precision of 0.07 and a recall of 1.00, resulting in an imbalanced F1-score of 0.14. It also faced challenges identifying Class 4 (TLSv1.2) with a precision of 0.19 and a recall of 0.02, indicating weaknesses in handling certain classes, especially when the data were imbalanced.

After applying 10-fold cross-validation, the NB model using encoding showed a slight drop in precision and F1-score compared to the single training–test split. The recall remained at 0.75, while the accuracy was still close to 0.75. Cross-validation provides a more reliable estimate of model performance on unseen data by averaging the results across different dataset splits.

The NB model using hashing showed an increase in accuracy to 83%, a notable improvement over the initial model's results. Although the macro average precision and recall slightly decreased to 0.27 and 0.40, respectively, there were improvements in the identification of key classes, such as Class (ICMP), with a precision of 0.41 and a recall of 1.00. Furthermore, the recall for Class (TCP) increased from 0.90 to 0.99, raising the F1-score from 0.86 to 0.91. For certain classes, such as TLSv1.3, the NB model using hashing still exhibited weak results. The higher overall accuracy and improved handling of major class distributions indicate that the enhanced model offers better generalization and efficiency than the original.

With 10-fold cross-validation, the performance of the improved NB model decreased slightly, with recall dropping to 0.75 (from 0.83), F1-score decreasing to 0.70, and accuracy falling to 0.75. This drop in performance suggests that the new model may not be as consistent when exposed to different parts of a dataset.

Table 1: Experimental results for network classification using encoding teaching and improved NB using the hashing technique

Methods	Precision	Recall	F1-score	Accuracy
Naive Bayes	0.72	0.76	0.73	0.76
10-Fold Naive Bayes	0.65	0.75	0.66	0.75
Improved Naive Bayes	0.72	0.83	0.76	0.83
10-fold improved Naive Bayes	0.72	0.75	0.70	0.75

Additionally, hashing techniques performed well when applied to the deep learning approach (Table 2), with the CNN model demonstrating high classification performance, especially in precision and recall, while the RNN model showed competitive results. The higher precision of the RNN model suggests that it may be more effective in identifying positive cases, whereas the higher recall of the CNN model indicates that they are better at capturing all relevant instances in a dataset. Therefore, CNNs may offer superior accuracy and recall, but RNNs can still be a viable alternative depending on the specific requirements of the application. In summary, a comparison of CNN and RNN models showcases the strengths of each approach.

Table 2: Experimental results for network classification using CNN and RNN models

Methods	Precision	Recall	F1-score	Accuracy
CNN	0.69	0.83	0.75	0.83
RNN	0.82	0.76	0.78	0.76

6. Discussion

The findings of this study align with the general trend in the related literature of improved classification accuracy through methodological enhancements. For example, Moore and Zuev (2005) reported an initial accuracy of 65.26% using an NB model, which was subsequently improved to 93.50% and 94.29% through the integration of kernel density estimation and FCBF, respectively. This demonstrates that the performance of NB models can be significantly improved through advanced feature selection and enhancement techniques.

Moreover, the performance of the enhanced NB model in this study is on par with the performance of alternative machine learning algorithms. A study comparing various algorithms, including C4.5, SVM, and RF (Jenefa and Moses 2018) found that C4.5 achieved superior performance, especially when coupled with CFS feature selection. Similarly, Sun et al. (2018) demonstrated that SVM models with enhanced feature selection techniques achieved higher classification accuracy. Thus, the 83% accuracy of the enhanced NB model presented here indicates that it is a viable option for network traffic classification.

In the context of deep learning applications, recent studies have shown significant performance improvements over traditional machine learning approaches. The use of techniques such as CNNs and SAEs (Lotfollahi et al., 2020) has yielded high accuracy rates in network traffic classification, with some frameworks achieving state-of-the-art performance. However, our enhanced NB model offers a less computationally intensive alternative, making it accessible to applications where resources may be limited.

7. Conclusion

In conclusion, the growth of internet usage has increased the need for efficient and accurate network traffic classification to detect malicious activities and unauthorized access. Traditional machine learning methods, such as NB, often struggle with the complexity of string-based categorical data, particularly IP addresses. To address this, we proposed the use of hashing techniques to convert such data into numerical form, simplifying processing and improving classification performance. Our findings demonstrate that the improved NB model, enhanced through hashing, achieves significantly better accuracy and recall than the standard version. While some challenges remain in handling certain classes, the overall results highlight the effectiveness of this approach in improving classification accuracy and efficiency in network traffic analysis.

While the improved NB model presents substantial advancements over its baseline counterpart, the landscape of machine learning and deep learning techniques in network traffic

classification continues to evolve. The model's performance can be further optimized through ongoing research and integration with more sophisticated algorithms and feature-selection methodologies. The observed improvements reinforce the notion that traditional models, when enhanced, can still hold significant relevance in the context of contemporary machine learning applications.

List of abbreviations

NB	Naive Bayes
CNN	Convolutional neural network
RNN	Recurrent neural network
FTP	File transfer protocol
DNS	Domain name system
SVM	Support vector machine
RF	Random forest
CFS	Correlation-based feature selection
ISVM	Incremental support vector machine
kNN	k-nearest neighbors
SAE	Stacked autoencoder
MLP	Multilayer perceptron
IDS	Intrusion detection system
MD5	Message digest 5
TP	True positive
FP	False positive
TN	True negative
FN	False negative
CSV	Comma-separated values
ICMP	Internet control message protocol
TCP	Transmission control protocol
TLS	Transport layer security
ROC	Receiver operating characteristic
FCBF	Fast correlation-based filter

Acknowledgment

This research has been funded by the Scientific Research Deanship at the University of Ha'il, Saudi Arabia, through project number BA-2207.

Compliance with ethical standards

Conflict of interest

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

References

Azab A, Khasawneh M, Alrabaee S, Choo KKR, and Sarsour M (2024). Network traffic classification: Techniques, datasets, and challenges. Digital Communications and Networks, 10(3): 676-692. https://doi.org/10.1016/j.dcan.2022.09.009

Cao J, Wang D, Qu Z, Sun H, Li B, and Chen CL (2020). An improved network traffic classification model based on a support vector machine. Symmetry, 12(2): 301. https://doi.org/10.3390/sym12020301

Chi L and Zhu X (2017). Hashing techniques: A survey and taxonomy. ACM Computing Surveys (CSUR), 50(1): 11. https://doi.org/10.1145/3047307

Davis J and Goadrich M (2006). The relationship between precision-recall and ROC curves. In the 23rd International Conference on Machine learning, Association for Computing Machinery, Pittsburgh, USA: 233-240. https://doi.org/10.1145/1143844.1143874 **PMCid:PMC3242122**

de Menezes NAT and de Mello FL (2021). Flow feature-based network traffic classification using machine learning. Journal of Information Security and Cryptography (Enigma), 8(1): 12-16. https://doi.org/10.17648/jisc.v8i1.79

Guerra JL, Catania C, and Veas E (2022). Datasets are not enough: Challenges in labeling network traffic. Computers and Security, 120: 102810. https://doi.org/10.1016/j.cose.2022.102810

Hancock JT and Khoshgoftaar TM (2020). Survey on categorical data for neural networks. Journal of Big Data, 7: 28. https://doi.org/10.1186/s40537-020-00305-w

Jenefa A and Moses MB (2018). An upgraded C5.0 algorithm for network application identification. In the 2nd International Conference on Trends in Electronics and Informatics, IEEE, Tirunelveli, India: 789-794. https://doi.org/10.1109/ICOEI.2018.8553826

Lewis DD (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In: Nédellec C and Rouveirol C (Eds.), European conference on machine learning: 4-15. Springer, Berlin, Germany. https://doi.org/10.1007/BFb0026666

Liu C, He L, Xiong G, Cao Z, and Li Z (2019). FS-Net: A flow sequence network for encrypted traffic classification. In the IEEE Conference on Computer Communications, IEEE, Paris, France: 1171-1179. https://doi.org/10.1109/INFOCOM.2019.8737507

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

Maxwell P, Alhajjar E, and Bastian ND (2019). Intelligent feature engineering for cybersecurity. In the IEEE International Conference on Big Data, IEEE, Los Angeles, USA: 5005-5011. https://doi.org/10.1109/BigData47090.2019.9006122

Moore AW and Papagiannaki K (2005). Toward the accurate identification of network applications. In: Dovrolis C (Ed.), International workshop on passive and active network measurement: 41-54. Springer, Berlin, Germany. https://doi.org/10.1007/978-3-540-31966-5_4

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

Nadler A, Aminov A, and Shabtai A (2019). Detection of malicious and low throughput data exfiltration over the DNS protocol. Computers and Security, 80: 36-53. https://doi.org/10.1016/j.cose.2018.09.006

Oloyede AA, Faruk N, Noma N, Tebepah E, and Nwaulune AK (2023). Measuring the impact of the digital economy in developing countries: A systematic review and meta-analysis. Heliyon, 9(7): e17654. https://doi.org/10.1016/j.heliyon.2023.e17654 **PMid:37501966 PMCid:PMC10368767**

Sharafaldin I, Lashkari AH, and Ghorbani AA (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In the 4th International Conference on Information Systems Security and Privacy, Funchal-Madeira, Portugal: 108-116. https://doi.org/10.5220/0006639801080116

Stallings W and Brown L (2015). Computer security: Principles and practice. Pearson, London, UK.

Sun G, Chen T, Su Y, and Li C (2018). Internet traffic classification based on incremental support vector machines. Mobile

Networks and Applications, 23: 789-796. https://doi.org/10.1007/s11036-018-0999-x

Wang P, Ye F, Chen X, and Qian Y (2018). Datanet: Deep learning based encrypted network traffic classification in SDN home gateway. IEEE Access, 6: 55380-55391. https://doi.org/10.1109/ACCESS.2018.2872430

Wang Z (2015). The applications of deep learning on traffic identification. BlackHat, USA. https://doi.org/10.54097/hset.v39i.6689