Maitin *et al. Journal of Big Data* (2025) 12:277
https://doi.org/10.1186/s40537-025-01183-w

Journal of Big Data

SURVEY

Open Access

Application of natural language processing techniques to network traffic processing for classification using deep learning models

Ana M. Maitin[1*], Carlos Arranz-Luque[2], Emilio Alba[2] and Álvaro J. García-Tejedor[1*]

*Correspondence:
Ana M. Maitin
a.maitin@ceiec.es
Álvaro J. García-Tejedor
a.gtejedor@ceiec.es
[1]CEIEC, Universidad Francisco de Vitoria, Madrid 28223, Spain
[2]FOQUM, Agustín Durán 24, Madrid 28028, Spain

Abstract

Background The rapid growth of encrypted network traffic has increased the need for effective and unbiased Network Traffic Classification (NTC). Traditional techniques struggle with encrypted data, limited feature availability, and high traffic volume, reducing their reliability in real-world scenarios.

Methods We propose a novel pre-processing methodology that analyzes raw network traffic into a textual format (nt2txt), enabling the application of Natural Language Processing (NLP) and Deep Learning techniques. This approach eliminates bias from protocol metadata, structures the data into fixed-size semi-flows, and uses rigorous data-splitting to prevent flow overlap between training and testing. An LSTM-based model is then trained to classify traffic using only payload data.

Results This work provides a scalable, protocol-agnostic framework for encrypted traffic classification, demonstrating the effectiveness of NLP techniques in improving model performance and reducing dataset bias. Our methodology achieved 88,87 ± 0,04% accuracy on a blind external dataset, outperforming similar LSTM and hybrid CNN-LSTM models. Metrics such as Cohen's Kappa and Matthew's Correlation Coefficient further confirm the robustness and generalizability of our approach.

Keywords Artificial intelligence, Cybersecurity, Traffic classification, PLN

Introduction

Background

The rapid increase in network traffic has made network management significantly more complex. Understanding and analyzing how information flows through computer networks is now essential. Many key tasks (such as ensuring security, detecting intrusions, providing Quality of Service (QoS), monitoring performance, allocating resources, and managing traffic) depend on effective traffic classification [1]. However, Network Traffic Classification (NTC) has become more challenging due to the growing number of connected applications and devices. These generate massive amounts of data, posing several difficulties:

- The diversity of protocols and formats complicates the extraction of features and relationships in the generated traffic.
- The dynamic component associated with network traffic involves changes in its topology, such as the addition or removal of devices. These changes in network configurations can impact the capture and analysis of network traffic.
- The massive volume generated in small time intervals in real network traffic, besides posing a technical limitation, can lead to the generation of noise and an excess of irrelevant data that hinders analysis and information extraction.

NTC has previously been addressed using methodologies such as port inspection [2], which associates applications or services with specific ports. However, the use of random ports by some programs or services renders this method ineffective [3]. Deep Packet Inspection (DPI) [3], involves analyzing the data being transported by each packet in search of patterns that links to a service or application. Nonetheless, this method has become unfeasible due to packet encryption. Consequently, Surface Packet Inspection (SPI) [3], has been proposed, focusing on analyzing packet headers or protocols. Nevertheless, SPI has proved to be imprecise and costly for handling large volumes of network traffic.

Overcoming these complications requires advanced analysis techniques, appropriate tools, and an adaptable approach to obtain precise and meaningful results. Consequently, Artificial Intelligence (AI) techniques have begun to be applied, as they have proven to be efficient in handling large data volumes and delivering high performance in feature analysis to draw insights from data in other fields [4]. Due to their significant potential, these techniques can offer efficiency and scalability, providing real-time processing capabilities.

Machine Learning (ML), defined by Arthur Samuel as "the field of study that gives computers the ability to learn without being explicitly programmed" [5], plays a key role in identifying patterns within large datasets. One of its main strengths is the ability to be easily retrained with new data, making it well-suited for network traffic analysis [3]. ML techniques are increasingly used in this field to support tasks such as quality of service management, trend monitoring, and security [6].

Among the AI models dedicated to sequential data analysis, Deep Learning (DL) techniques represent the latest advancements in NTC [7, 8]. DL techniques have demonstrated a superior ability to abstractly analyze complex and noisy data, even learning hierarchical representations and intrinsic data features. This allows for capturing more sophisticated and subtle relationships, ultimately enhancing model performance and expanding their range of applications, as seen in examples like Stable Diffusion, AlphaFold, and ChatGPT [9–11]. In particular, techniques such as Recurrent Neural Networks (RNN) [12], specifically the Long Short-Term Memory (LSTM) architectures, and Natural Language Processing (NLP) techniques [13], based on Transformer models [14], are at the forefront of NTC. They are renowned for their robustness in handling class imbalance [15]. Both of these techniques are designed to uncover relationships within unstructured data, extracting inherent properties and connections.

Increasingly complex methods are being adopted through the use of modern technologies. Studies such as [16] and [17] highlight both the limitations of traditional approaches and the potential of Deep Learning techniques. In [16], a multi-level classification framework is proposed that combines 1D-CNN, attention-based Bi-LSTM, and

stacked auto-encoders to extract complex spatial and temporal features from encrypted traffic. Additionally, it addresses class imbalance through a data augmentation strategy based on GANs. In [17], the focus is on real-time encrypted traffic classification, emphasizing the challenges of processing high-speed data streams under resource constraints.

In a literature review by [3, 18], various models employed for network traffic analysis are discussed. While ML algorithms like Support Vector Machines (SVM), Decision Trees (DT), and k-Nearest Neighbors (KNN) have been used, there is a prevailing trend towards the adoption of DL techniques. This inclination is due to the fact that more complex techniques reduce the need for extensive preprocessing and allow for the extraction of information from raw data. They also deliver superior performance and are better equipped to handle substantial datasets, thereby favoring potential real-time traffic analysis applications. Among these models, the most frequently used one is the Multi-Layer Perceptron (MLP), appearing in 13 out of the 28 studies reviewed. The Convolutional Neural Network (CNN) follows with a frequency of 9 articles. Additionally, two hybrid models based on CNN + LSTM were considered [19], and only two studies utilized RNN, employing LSTM layers [20, 21]. The metrics obtained with RNN models resulted in the hit rate or accuracy = $73{,}64\%$, harmonic mean of the precision an recall or F-measure = $59{,}53\%$, geometric mean or Gmean = $73{,}31\%$ in [20], and accuracy = $83{,}69\%$, F-measure = $77{,}04\%$ in [21].

However, it should be noted that the data preprocessing procedures were not consistent across the reviewed studies, revealing a lack of standardized methodology in this domain. This inconsistency poses a significant challenge when comparing the performance of different models, as variations in data handling can greatly influence the results. More critically, non-uniform preprocessing can introduce hidden biases into the datasets (particularly when features such as IP addresses, port numbers, or protocol metadata are not properly anonymized or filtered). These elements can inadvertently create correlations between the training and testing sets, allowing models to learn shortcuts based on these artifacts rather than on the actual content or behavior of the traffic. As a result, the model may demonstrate high accuracy during validation but fail to generalize in real-world or unseen scenarios. This kind of bias undermines the reliability and robustness of the model, making its performance metrics misleading. Therefore, adopting a consistent and bias-aware preprocessing methodology is essential to ensure fair evaluation and enhance the generalizability of AI-based approaches in NTC.

In this paper, we propose a data pre-processing methodology that takes into account the volume of real traffic and class imbalance, while also avoiding bias introduced by correlations among similar data in different stages of model training/validation/testing in ML models. It is crucial to establish a standardized protocol for pre-processing network traffic data, as ML and DL models are increasingly being used in the field of cybersecurity to offer improved network management, threat detection, and application performance optimization. To achieve this, we must identify packets belonging to the same flow, which may share information, as an inadequate representation of the temporal sequence of events can introduce elements that yield correlations, biasing training results and impeding model generalization [1].

However, we have not found a rigorous and detailed methodology that could serve as a guide for developing this pre-processing. Therefore, we propose a method that eliminates any bias by conducting the analysis on raw content, making it independent of the

device connected to the network. To facilitate the extraction of features or relationships among the different packets composing a flow, we propose a pre-processing based on NLP techniques by considering traces as a structured sequence of events, establishing an equivalence between Network Traffic (NT) and Text.

To facilitate the objective evaluation of the results obtained from NTC, we have conducted an analysis using various metrics employed in multi-class problems. This approach allows for effective measurement and comparison of the model's performance in classifying multiple categories, utilizing metrics such as the confusion matrix [22], Cohen's Kappa score [23], Matthew's correlation coefficient [24], and log loss [25]. Evaluating results across different metrics contributes to a deeper understanding of the model's performance in multi-class classification problems, aiding in making informed decisions to enhance overall system accuracy and performance. Implementing these metrics will provide us insights into the performance of a model, built upon the most commonly used ones in the literature, in an NTC problem and allow us to validate the proposed methodology.

Therefore, this document presents a rigorous methodology for preprocessing network traffic data to enable the use of NLP techniques in real-world traffic scenarios. The approach supports parallel processing across multiple devices, making it suitable for handling the high data volumes found in 4G and 5G networks. This work lays the foundation for researchers in cybersecurity to apply more advanced methods aimed at improving network management and security.

This paper is organized as follows: In the Methods section, we present the dataset alongside the proposed methodology, highlighting potential biases that may arise during the preprocessing of PCAP files at each stage. We provide a detailed account of the implementation of this process, the methods employed for result evaluation, and introduce metrics tailored for multiclass problems. Moving on to the Results section, we apply these preprocessing techniques to a model founded on LSTM layers, whose architecture has been meticulously optimized. In the Discussion, we benchmark the results obtained with the proposed method against some state-of-the-art models in trace classification. Finally, the Conclusions section encapsulates a summary of the most noteworthy findings stemming from this study.

Methods

Dataset

Network traffic consists of data packets that are sent and received between electronic devices connected to the network. These packets contain fragments of the communication they are part of, which can take various forms, such as file transfers, emails, website requests, instant messages, video conferencing, and any other type of communication. Each packet contains multiple information fields that identify different attributes, such as their source, destination, or communication protocol type, in addition to the data being transmitted, that is the payload. These fields are recorded in PCAP files (Packet Capture), which can be obtained using network sniffers, dedicated programs for the continuous logging of all network packets.

The dataset used in this work bears the identifier code "ISCXVPN2016" and is sourced from the Canadian Institute for Cybersecurity, University of New Brunswick [26]. Access

to this dataset is available via the following link[1]. The dataset has been previously validated by other research in the field [3], demonstrating that the contained data traffic is sufficient for conducting the classification task.

This dataset captures traffic from 5 categories: chat, file transfer, streaming, Voice over IP (VoIP), and email. Each category records the traffic associated with different applications, specifying the total number of registered PCAP files, total packet count, and recording time in minutes. The dataset consists of 109 labeled PCAP files containing a total of 21.910.108 packets. Table 1 summarizes these characteristics for each category and the overall item count.

As seen in Table 1, we have an imbalanced dataset biased toward the "File Transfer" class. It is essential to note that while any type of communication generates multiple network packets, the transfer of files generates a higher volume of packets due to its heavier nature compared to other forms of communication. Conversely, communication methods like email contain less information, resulting in a lower volume of generated packets. Therefore, the model evaluation needs to consider this class imbalance to ensure an optimal learning process and accurate result generalization.

Given the nature of the dataset and its inherent imbalances, a robust and carefully designed preprocessing methodology becomes essential to ensure fair and meaningful model evaluation.

Proposed pre-process methodology

To ensure proper data processing and eliminate potential biases affecting model predictions, it is crucial to note that network traffic logging involves capturing various fields identifying packets belonging to the same communication flow. Consequently, classifying network traces directly using raw data might lead to correlations in model outcomes, limiting their generalization and hindering real-time analysis applicability. Therefore, it is essential preprocessing these data, employing diverse metrics to assess results, and adhering to a rigorous validation process. The proposed methodology consists of two parts: the first one involves data preprocessing, and the second one establishes an equivalence between Network Traffic and Text (nt2txt).

Data pre-processing

In our approach to NTC, we aim to use only the raw content of the transmitted information. To achieve this, we remove all fields from the PCAP file except the payload, which is the field that contains the data carried by the packet. Packets with empty payloads are

Table 1 Summary of the number of applications, PCAP files, total packages and logging time for each category in the ISCXVPN2016 dataset

Categories	Number of applications	Number of PCAP files	Number of packages	Recording time (min)	Percentage of information
Chat	6	20	153.451	164,12	0,19%
File transfer	4	32	11.007.165	190,88	38,66%
Streaming	7	30	4.991.536	294,65	30,08%
VoIP	4	23	5.682.877	726,23	31,04%
Email	1	4	75.079	207,45	0,03%
Total	22	109	21.910.108	1583,33	100%

[1] https://www.unb.ca/cic/datasets/vpn.html.

considered null data. These packets are also removed, as they can introduce noise into the model.

The methodology presented in this document is designed for potential real-time implementation, ensuring that packet processing time is optimized to minimize delays, especially when analyzing large volumes of real-world data. To achieve this, captured data must be fed into the model without waiting for the entire communication to end. At the same time, the number of packets must be limited to ensure optimal model performance. This strategy reduces classification delay and ensures reliable results from the trained model. Additionally, it is essential to note that NT is composed of flows with a variable number of packets, resulting in an imbalanced dataset. This imbalance is a significant challenge in encrypted traffic classification [19], as it can lead to reduced classifier performance or facilitate the concealment of malicious cyberattacks within vast amounts of normal data.

The designed methodology addresses this issue by restricting the input data to the model and implementing a subdivision into semi-flows, which are subsets of the same flow containing a fixed number of packets. This subdivision allows us to standardize the size of the input samples, ensuring that each semi-flow contributes equally to the learning process, regardless of the total number of packets in the original flow. By doing so, the methodology prevents classes with longer flows (such as file transfers) from dominating the training process due to their higher packet volume. At the same time, it ensures that shorter flows (such as those from emails or VoIP) are also adequately represented. This strategy reduces class imbalance and enables the model to learn more balanced patterns across different traffic types, improving both classification accuracy and generalization to real-world scenarios.

Determining the size of each semi-flow requires setting two conditions that act as processing hyperparameters. These parameters must be optimized depending on the characteristics of the classification model used. First, the number of packets that will form each semi-flow is defined. Second, a maximum duration for the semi-flow is set, based on the time difference between the first and last packet included. Both hyperparameters directly affect the data processing time and the model's prediction latency. Adjusting them appropriately helps balance real-time processing constraints with model performance.

Packets are added to a semi-flow until either the designated number of packets or the maximum duration is reached. If the semi-flow completes without surpassing the set time, it is accepted. Conversely, if the maximum duration is reached before completing the semi-flow, the packets within it are discarded, and a new semi-flow construction begins.

nt2txt equivalence

Once the pre-processing of data has been completed, an equivalence between NT and text is established. In this approach, the data transported within a semi-flow is identified as text, wherein the content of each packet would be akin to a sentence. Extending this analogy, we delineate the fundamental units that comprise these sentences, which are akin to words.

To carry out this association, we need to introduce NLP techniques, that is the tokenization. This is the standard methodology within text processing techniques, responsible

for segmenting sentences into elementary processing units called tokens. To perform this step, the payload, which consists of a sequence of variable-length hexadecimal characters, is identified and segmented into fixed-length intervals, conducting tokenization. The token size is one of the hyperparameters that needs to be optimized. This allows us to identify each segment or token as a word within the text that comprises the NT. Figure 1 illustrates this process.

Similar to text processing in the field of NLP, the end of each packet is marked by introducing an end-token. In text processing, this step is added to signal the model when the end of a sentence, a paragraph, or text has been reached. This end-token is distinctively identified to ensure the analyzing model does not confuse it with another token representing a word. For our methodology, the end-token corresponds to the highest value within the hexadecimal configuration, denoted by the character "F," repeated as many times as the token components. Subsequently, each token is converted to decimal format, aiming to obtain a vectorized (similar to embedding in NLP) representation to feed into the AI model. This step is performed after tokenization to prevent fragmenting the information corresponding to a single hexadecimal character.

Given the variable length of the payload, we need to establish a specific value that would correspond to the length of the sentence, which is another hyperparameter that must be optimized and adapted to the capabilities of the model intended for subsequent classification. Finally, the outcome of this preprocessing is a set of semi-flows containing packets whose information is segmented using tokens.

In this context, the analogy between network traffic and textual data can be described as follows: a complete flow is akin to an unordered text, which we organize into thematic units or semiflows. Each semiflow represents a coherent section of the text, consisting of sentences, analogous to packets. These sentences (packets) are then tokenized, breaking down their content into discrete units similar to words. Once tokenized, the semiflows are labeled accordingly to enable supervised learning during the model training phase.

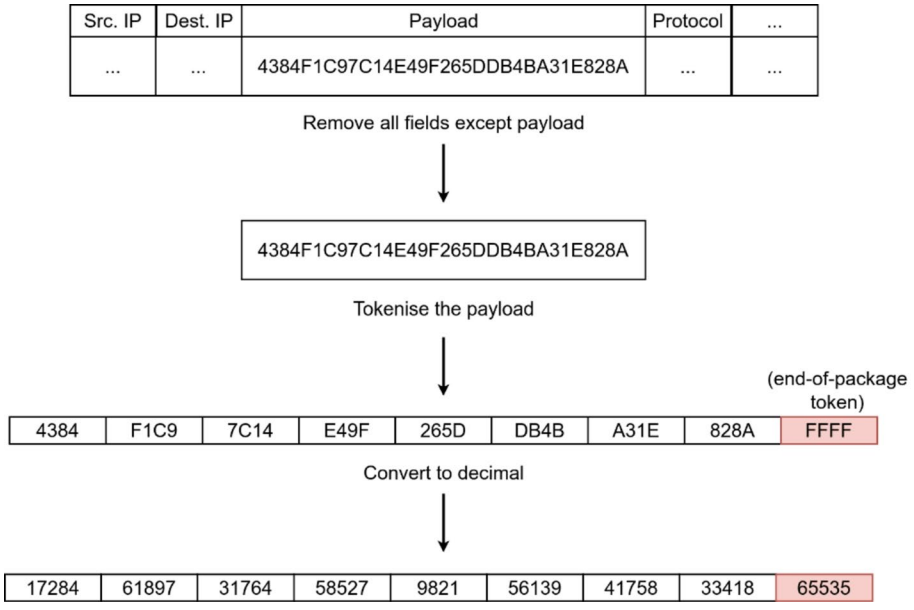

Fig. 1 Payload tokenization process to perform nt2txt traffic equivalence

The objective of carrying out the nt2txt equivalence is to aid the model in information extraction, promoting the exploration of relationships between the tokens comprising the payload of a packet and among the different packets forming a semi-flow. This approach aims to identify characteristic patterns of each communication category, facilitating their identification.

Implementation of the methodology

Next, we detail how the proposed methodology could be implemented to feed a DL model. To do this, we divide the implementation of the methodology into the steps described in the previous section, namely data preprocessing and nt2txt.

Implementation of data pre-processing

We initiated the pre-processing of the PCAP files by converting them to.CSV files using tshark, the Command-Line Interface (CLI) of Wireshark, to streamline data handling. The specifics of the information contained in the files are detailed in Table 2.

The generated CSV files were stored in a folder structure labeled for ease of information retrieval, particularly if there was a need to identify a single file. Upon loading these files, they were transformed into DataFrames using pandas[2] library to facilitate data handling. Each row within the DataFrame represented one of the captured packets, while the columns corresponded to the information specified in Table 2.

Next, a dictionary was created, where the keys corresponded to each of the categories contained in the dataset, which were five in our case. The content stored under each key was a list comprising the DataFrames belonging to that specific category. This method creates an organized data structure, as depicted in Fig. 2, enabling easy modification and swift extraction of desired information in an efficient manner.

Following this, the "Hash" column was added to each DataFrame, containing unique identification characteristics of the communication or flow to which each packet belonged, enabling subsequent identification. To accomplish this labeling, a hash function was applied to the columns IPsrc, IPdst, Protocol, TCPsrcport, TCPdstport, UDPsrcport, and UDPdstport, concatenating the characters they contain. Subsequently, the columns utilized by the hash were removed, as they might introduce biases during

Table 2 Displays the categories exported to.CSV along with a brief description of each field

Item	Description
Nº	ID number of the captured packet
Time	Timestamp in wich the packet is captured within the recording of the PCAP file
Source	Source device IP
Destination	Destination device IP
Protocol	Network protocol present in the captured packet
Length	Length in bytes of the captured packet
tcp.srcport	Source port in a TCP communication
tcp.dstport	Destination port in a TCP communication
tcp.len	Length in bytes of the data field in a TCP communication
udp.srcport	Source port in an UDP communication
udp.dstport	Destination port in an UDP communication
tcp.payload	Contains the data transported via the TCP protocol
udp.payload	Contains the data transported via the UDP protocol

[2] https://pandas.pydata.org/.

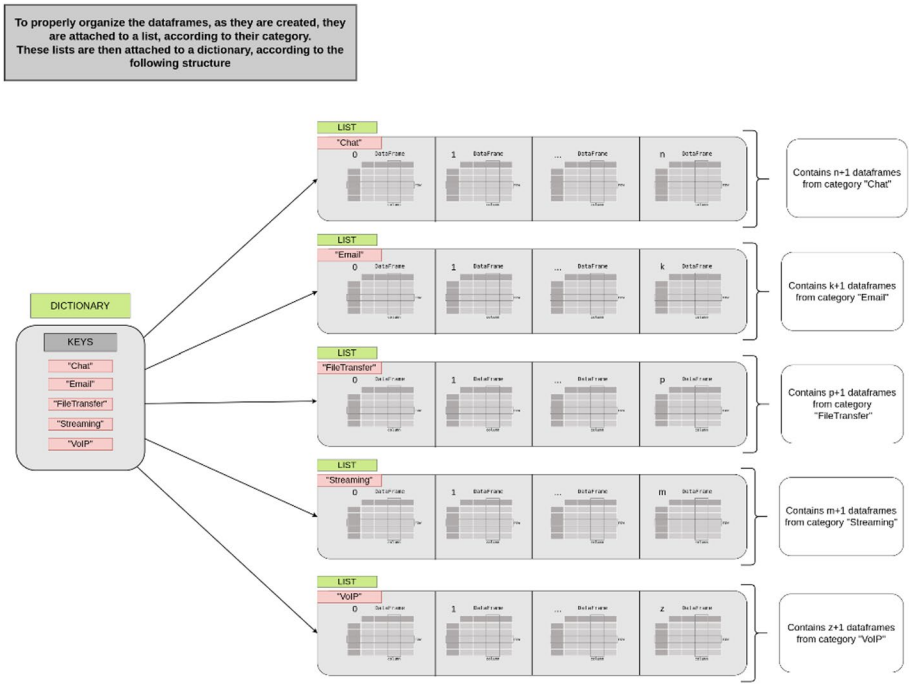

Fig. 2 Diagram of the dictionary structure, associating a list of DataFrames to each key

training. Additionally, the packet content was evaluated to eliminate those that did not provide information and could impair the model's performance. The exclusion criteria for packets were as follows:

- **Packets that, according to their protocol, did not contain information.** The dataset was carefully reviewed to identify all transport protocols present. Based on this inspection, the following protocols were selected for further analysis: "TCP," "UDP," "SSL," "SSLv2," "TLSv1," "TLSv1.1," "TLSv1.2," "TLSv1.3," "SSHv2," "GTP < TCP>," "GTP < UDP>," "GTP < QUIC>," "GTP < SSL>," "GTP < SSLv2>," "GTP < TLSv1>," "GTP < TLSv1.1>," "GTP < TLSv1.2>," "GTP < TLSv1.3>." The selection was restricted to protocols that contain relevant data within their payload in a communication. Protocols that do not provide such information, like ARP, were excluded to avoid introducing noise into the models. For this dataset, the specified transport protocols were TCP and UDP.
- Packets with an empty payload field
- **Packets whose data had a length** $\leq$ 2**characters** as they might be associated with acknowledgment confirmations.

Following this processing step, the Length and TCPlen columns were removed since the objective was to classify the transported data. The structure of the resulting DataFrames, which form the class structure, can be observed in Fig. 3.

The next step involved extracting flows and semi-flows. The "Hash" column in each of the processed DataFrames allowed us to identify the different flows contained in the derived PCAP file, since a flow consists of a set of packets with the same hash. This method enables the identification of the packets corresponding to the same communication in each capture, regardless of their appearance order. This process prevents packets belonging to the same flow from being considered in both the training and

	Time	TCPpayload	UDPpayload	Hash
0	2.336035	NaN	48f84a1d4e0b089813a5d193e496363d96f23f8f424952...	f22c891361a2bcc2ca93c23104632357
1	2.336072	NaN	48f84a1d4e0b089813a5d193e496363d96f23f8f424952...	f22c891361a2bcc2ca93c23104632357
2	2.336354	NaN	54f84a1d4e0b089813d5647e1bdcd706ac651e67847c8a...	f22c891361a2bcc2ca93c23104632357
3	2.336367	NaN	54f84a1d4e0b089813d5647e1bdcd706ac651e67847c8a...	f22c891361a2bcc2ca93c23104632357
4	2.336582	NaN	4ef84a1d4e0b08981379de027e2b0b60000b537d248e4f...	f22c891361a2bcc2ca93c23104632357
...	...	...	...	...
96099	32.698808	NaN	4bcff6034eac4d18657bf4ccca1dc8cfaa1d955ce620ce...	73b8a303944602ec5f1d5c58efcf728b
96100	32.698811	NaN	4bcff6034eac4d18657bf4ccca1dc8cfaa1d955ce620ce...	73b8a303944602ec5f1d5c58efcf728b
96101	32.698901	NaN	43cff6034eac4d1865a84253c71dc5a0576964e70e24a4...	73b8a303944602ec5f1d5c58efcf728b
96102	32.698905	NaN	43cff6034eac4d1865a84253c71dc5a0576964e70e24a4...	73b8a303944602ec5f1d5c58efcf728b
96103	32.704882	NaN	475108e200003f519a33e86ea45ddb62ee74bc04dabe37...	66901cc48356c7be7eed849981212f09

Fig. 3 Final structure of the DataFrame after processing and removal of data that could introduce bias into the model

validation sets, thus avoiding correlations in the results. Once the flows were identified, we excluded those with an insufficient number of packets to enhance the model's learning. Hence, the minimum number of packets per flow is a preprocessing hyperparameter that will depend on the chosen model.

Once the flows were extracted, we proceeded by removing the "Hash" column, as well as the empty "Payload" column (either tcp.payload or udp.payload), since each flow corresponded to a single protocol. As a result, the DataFrame for each flow contained only two columns: "TimeStamp" and "Payload." However, when considering real-time applications, it's important to note that each flow can contain a large number of packets. This increases the complexity of the AI models and can complicate obtaining timely results. To address this issue, we divided each flow into smaller subsets called semi-flows, based on the following criteria:

- Semi-flows are comprised of sequential packets extracted from the same flow based on the TimeStamp column.
- Each semi-flow consists of a specified number of packets, denoted as N, serving as a preprocessing hyperparameter.
- We limit the maximum time, denoted as 't', to record the packets forming the semi-flow using the TimeStamp, representing another preprocessing hyperparameter. This involves calculating the difference between the registration values of the first and last packet composing the semi-flow.

From a PCAP file or its associated CSV, we obtained a list of semi-flows, each containing a fixed number of packets, denoted as N, registered within a maximum time t, associated with the same communication flow. A diagram of this process can be seen in Fig. 4.

nt2txt implementation

Once we identified the flows and semi-flows, and the data was preprocessed, we could apply the equivalence between NT and text. To perform the tokenization of the payload of each packet contained within each semi-flow, we needed to establish two pre-processing hyperparameters:

- **Number of hexadecimal characters per packet**, or its text equivalent, which represents the phrase's length. After extracting the payload, it is truncated to a maximum length or padded with zeros to fill the remaining characters.

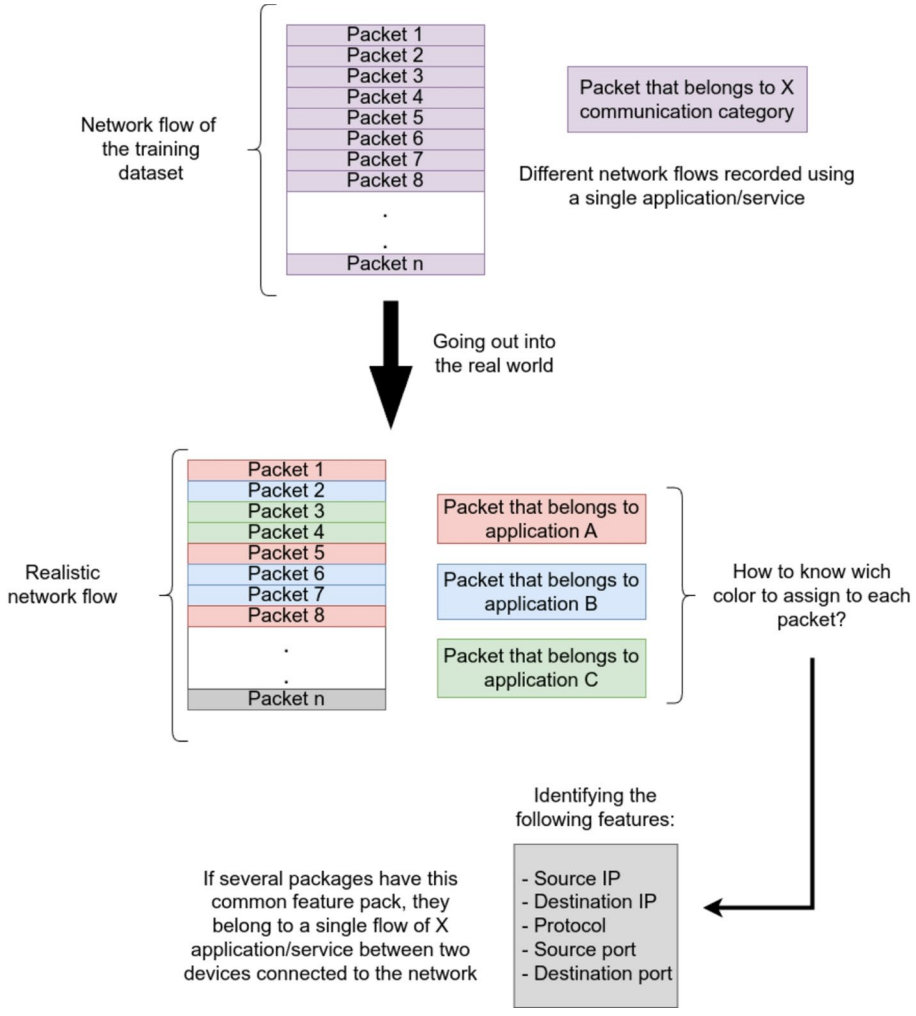

Fig. 4 Diagram of the described methodology to obtain flows and semi-flows

- **Number of hexadecimal characters composing a token.** The token's length determines the amount of information within each segment.

These hyperparameters enabled us to split each payload into a fixed number of tokens of the same size. Thus, each token contained an equivalent amount of information associated with the packet's data. To signal the end of a phrase or packet payload, we inserted an end-token composed solely of "F" characters. Subsequently, each hexadecimal token was converted into a decimal token through standard base conversion. This process transformed the information contained in each payload into a list of decimal tokens. Those lists associated with payloads within the same semi-flow were stored together in a new list.

Finally, a list was created containing these previous lists, associated with semi-flows, and simultaneously, another list with their corresponding classes or one-hot-encoded categories was generated. Both lists were ultimately transformed into numpy arrays (with numpy[3] library) for data and labels, used for the classification problem. The shapes of these resulting arrays were (S, N, T) and (S, C), where S represents the total number

[3] https://numpy.org/.

Table 3 Results of optimal hyperparameters for network trace preprocessing

Preprocessing Hyperparameters	Value
Minimum number of packets per flow	20
Number of packets per semi-flow	20
Maximum time between packets of a semi-flow (seconds)	30
Length of the payload of a packet	400
Number of characters per token	4

Table 4 Results of the optimal parameters for the model used based on LSTM layers for the above preprocessing parameters

Model Hyperparameters	Value
Batch size	10.000
Learning rate	10^{-4}
Number of LSTM layers	2
Hidden state dimensions	75
Number of dense layer neurons	100

of semi-flows, N is the total number of packets per semi-flow, T denotes the number of tokens per semi-flow (calculated as an integer quotient between the fixed payload length and the token length), and C represents the number of classes or categories.

Evaluating the proposed methodology

To validate the proposed processing methodology, we constructed a hybrid Recurrent Neural Network (RNN) model based on LSTM layers, followed by an MLP-based classification module for performing classification. Hyperparameter tuning was conducted, so that the results of this model could be compared with other RNN-based models from the literature.

Tuning was performed to find the best hyperparameters for both the preprocessing and the model used. The optimal configuration for the preprocessing is shown in Table 3:

Table 4 shows the optimal values obtained for the LSTM layer-based model:

In the considered model, each LSTM layer had the same hidden state. They were followed by a fully connected layer with 100 neurons and ReLU activation function, followed by another fully connected layer with 5 neurons and a Softmax activation function. Adam optimizer and CategoricalCrossEntropy loss function were used. Additionally, to prevent overfitting, an Early Stopping function was implemented to halt training once this trend was detected. This model consisted of 106.505 parameters, which were trained on Amazon Web Services (AWS), and the virtual machine specifications were: T4 GPU (16 GB VRAM), 16 GB RAM, 4 vCPUs Intel Xeon.

Although the dataset had been previously utilized, upon comparing its properties with real-world data traffic, we noticed a scarcity in both the number of classes and the volume of traffic. Therefore, in our quest for a rigorous methodology providing insights into model biases, we implemented a 5-fold Cross-Validation (CV). This validation method offers insights into how the model's performance varies concerning the selection of data. Additionally, before the CV process, we allocated 10% of the flows to create a blind test set (while maintaining the data percentage of each class), while the remaining flows were used in the validation process. This separation ensures data independence among the training, validation, and test processes, thus preventing potential correlations that could distort the model results.

Metrics

The model results were assessed using various metrics specific to multi-class problems, including:

- **Confusion matrix**: It displays predicted versus real values, enabling the identification of True Positives, True Negatives, False Positives, and False Negatives [22].
- **Cohen's Kappa score.** It measures the level of agreement between two assessors in a classification problem involving N items in K mutually exclusive categories [23]. It is defined by the equation:

$$\kappa = \frac{p_0 - p_e}{1 - p_e}, \tag{1}$$

where p_0 is the agreement rate between assessors, that is, $p_0 = (number\ of\ identically\ classfied\ items)/N$, and p_e is the probability of agreement if the assessors classify the samples randomly, that is,

$$p_e = \sum_{k=1}^{K} \frac{n_{1k}}{N} \frac{n_{2k}}{N}, \tag{2}$$

where n_{ik} is the number of items the assessor i classifies in category k. If the assessors completely agree, then $\kappa = 1$. If there is no agreement between the assessors other than what would be expected by chance, then $\kappa = 0$. It should be noted that the value of κ can be negative, indicating that there is no relationship between the scores of the two assessors, which may occur by chance or reflect a real trend.
In our case, we assume that the classification performed by the assessors corresponds to the real and predicted labels. Therefore, agreement between the actual and predicted labels is measured, taking into account possible hits by chance.

- **Matthew's Correlation Coefficient.** Measures the correlation between real and predicted labels in a classification problem of N items into K categories [24]. In fact, in the binary case, it corresponds to the Pearson correlation coefficient. It can be calculated as:

$$MCC = \frac{Nc - \sum_{k=1}^{K} t_k p_k}{\sqrt{N^2 - \sum_{k=1}^{K} t_k^2} \sqrt{N^2 - \sum_{k=1}^{K} p_k^2}}, \tag{3}$$

where c is the number of samples correctly classified, t_k is the number of real samples in category k, and p_k is the number of samples predicted in category k. MCC returns a value in the range [-1,1], where 1 represents a perfect prediction, 0 is no better than random prediction, and – 1 indicates complete disagreement between prediction and observation. In this case a high degree of correlation indicates a good performance of the model.

- **Log loss.** Measures the performance of a classifier where the predicted outcome is a probability value between 0 and 1. Log loss is calculated for each row using the Log loss equation, which measures the difference between the prediction and the true label. Then, the average Log loss across all rows in the test set is computed. More suitable classifiers have progressively smaller Log loss values; thus, the classifier with the smallest Log loss value exhibits better performance [25]. It is defined as:

$$L_{log}(y,p) = -(y\log(p) + (1-y)\log(1-p)) \qquad (4)$$

where the true label $y \in \{0,1\}$ and the estimated probability $p = Pr(y=1)$.

Results

The Fig. 5 displays the training curves for both accuracy and the loss function in one of the training processes of the CV.

The accuracy and loss curves obtained during training exhibit a consistent upward and downward trend, respectively. This behavior indicates that the model progressively improved its ability to correctly classify the input data while simultaneously minimizing the prediction error. The absence of divergence between the training and validation curves suggests that the model generalizes well to unseen data, providing evidence that overfitting was effectively avoided. Therefore, these learning curves validate the effectiveness of the training process and confirm the robustness of the proposed methodology.

Figure 6 presents the confusion matrix, which reveals that the model was affected by the class imbalance, as it failed to produce correct predictions for the minority classes (Chat and Email). This outcome suggests that the proposed methodology alone is not sufficient to address the challenges posed by imbalanced datasets, as it does not achieve adequate representation of minority classes. Additionally, the confusion matrix shows a significant proportion of Streaming traffic misclassified as File Transfer. Although this is technically an incorrect classification, the similarities between these two types of network traffic (particularly in the high number of generated flows) can reasonably explain the model's confusion.

Based on the confusion matrix results shown in Fig. 6, the weighted macro-F1 score can be calculated. The F1 scores obtained for each class are as follows: Chat = 0; Email = 0; File Transfer = 88,64; Streaming = 80,00; and VoIP = 98,13. The average of these values yields an overall F1 score of 53,35. However, it is important to note that each class has a different representation in the dataset. Therefore, a weighted F1 score provides a more accurate evaluation by incorporating each class's proportion within the dataset: Chat = 0,19%; Email = 0,02%; File Transfer = 38,65%; Streaming = 30,07%; and VoIP = 31,06%. When applying these weights, the final weighted F1 score is 88,80.

The training results showed no variations among the different 5-fold CV processes, suggesting that the model was not biased by the data selection. Therefore, the training was robust, indicating the potential for generalization of these results to real traffic

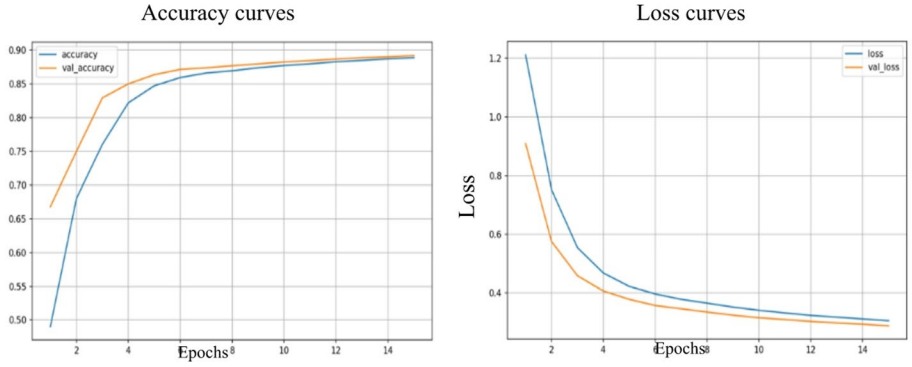

Fig. 5 Graphs showing accuracy and loss functions during the training and validation processes in one iteration of the cross-validation procedure

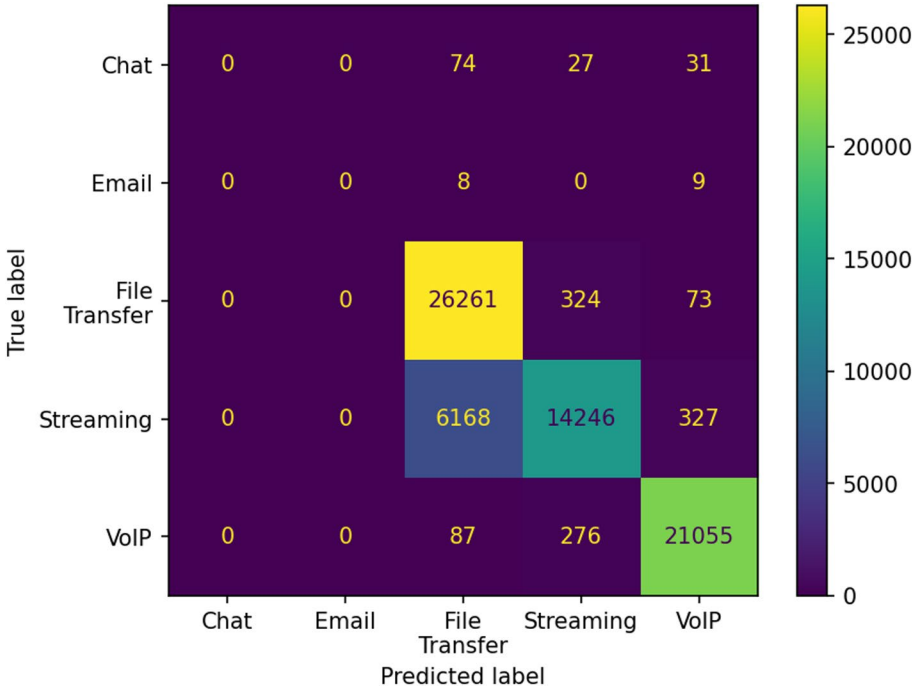

Fig. 6 Confusion matrix for the 5-class classification problem. The horizontal axis represents the predicted outcomes by the model, while the vertical axis displays the actual results from the test set

Table 5 Results of accuracy and loss metrics from the Cross-Validation process for training, validation and testing

Metrics	Train	Validation	Test
Accuracy (%)	88,39 ± 0,07	88,61 ± 0,30	88,87 ± 0,04
Loss	0,31 ± 0,01	0,29 ± 0,01	0,29 ± 0,00

Table 6 Metrics results for the Cross-Validation process for the test set

Metrics	Test
Accuracy (%)	88,87 ± 0,04
Log Loss	0,29 ± 0,00
Cohen kappa score (%)	83,02 ± 0,07
Matthews Correlation Coefficient	0,84 ± 0,00

analysis. This is supported by the metric outcomes demonstrating small standard deviation values, as presented in the results from Table 5.

Once the absence of bias in the model had been verified, the previously described metrics were evaluated on the blinded test set. The results of the metrics can be seen in Table 6.

The metrics results indicate that the proposed model achieved a high **accuracy** rate of $88{,}87 \pm 0{,}04\,\%$. However, given the imbalanced classes, it's necessary to evaluate the model's performance with consideration for prediction security.

Log Loss evaluates the difference between the prediction and the true label, where a value close to 0 indicates a better-performing classifier in the context of multi-class problems. Therefore, in this scenario, we have a well-performing classifier.

The **Cohen's Kappa** coefficient indicates the degree of agreement between predictions and real labels considering chance agreement. For the proposed model, a value of

Table 7 Results of the proposed model compared to other studies

Model	Input Data	Accuracy (%)
Our Model	Proposed preprocessing	88,87 ± 0,04
(20) LSTM	Raw/Packet level features	73,64 ± 1,56
(21) LSTM	Raw/Packet level features	81,64 ± 0,60
(19) CNN + LSTM	Packet level feature	96,32
(20) CNN + LSTM	Raw/Packet level features	77,95 ± 0,41
(21) CNN + LSTM	Raw/Packet level features	84,25 ± 0,49
(27) CNN + LSTM	Raw traffic/time series features	74,26 ± 0,98

$83{,}02 \pm 0{,}07\%$ was obtained, signifying a high agreement rate, thereby indicating the model's robustness and successful training process.

Lastly, the **Matthews Correlation Coefficient** value suggests a high correlation between real and predicted values.

Discussion

Given that there are other studies in the literature employing LSTM-based models for NTC, we will conduct a comparison of the results using Table 7.

When comparing the results of the proposed model with other LSTM-based models, we find that our methodology not only effectively prevents biases but also helps the model extract features from the majority class data, leading to improved classification results. It is worth noting that our LSTM model outperforms the current state of the art, even when using a similar base architecture. However, we wanted to compare our results with those of hybrid CNN + LSTM models to better assess the potential of our proposed methodology. As seen in the comparison, except for the model in [19], our model outperforms the current state of the art, including hybrid models, thanks to the preprocessing steps outlined. It is important to highlight that these studies used raw data without preprocessing. Thus, they did not eliminate biases caused by ports or IP addresses. Moreover, they did not address class imbalance. In the case of [19], the validation process is not described, and it is unclear whether the model was properly trained or if the bias introduced by correlations between different training sets was eliminated.

The assessment using different metrics has provided a deeper understanding of the generalization capacity of the proposed model to yield robust results free from biases, which would facilitate the implementation of a real-time proof of concept. Processing times are approximately 29 min for every 64 GB of loaded PCAP, averaging 2,2 GB of PCAP per minute. Considering that the 64 GB dataset took 26,35 h to record, we can conclude that the processing speed in a hypothetical real-time application would be quite optimal.

Among the limitations of the study, we must highlight that this model has shown sensitivity to class imbalance, as it was unable to correctly classify the minority classes. Although these classes, namely chat and email, represent only about 0,19% and 0,03%, respectively, of the total dataset, we chose to retain the imbalance in order to evaluate the generalization capacity of the proposed methodology in the developed model. This also allowed us to compare our results on equal terms with other methods. Therefore, it is necessary to apply techniques to address class imbalance alongside this methodology. Additionally, we recommend using a more powerful model that can tolerate some degree of imbalance between classes, minimizing the noise introduced into the model.

The proposed methodology is highly adaptable and not restricted to any specific device, communication protocol, or transmission medium (e.g., 4G or 5G), making it applicable across various network environments and scalable to different infrastructures. By focusing on key aspects like payload data and removing irrelevant fields, the preprocessing method addresses challenges such as noise reduction and packet inconsistencies, enhancing the model's generalization and real-time applicability.

For future work, we plan to introduce Transformer-based models to compare their performance with the current results and to scale the problem by incorporating a larger volume of data to better simulate real-world traffic. This will involve integrating new datasets and increasing the complexity of the data. Additionally, we aim to compare both approaches in a controlled real-world environment to conduct scalability tests with live network traffic. This will allow us to evaluate computational resources and assess precision metrics within this simulated environment. We also consider performing ablation studies on these models to strengthen the results and identify areas for improvement.

Conclusions

This study presents a novel methodology for encrypted network traffic classification by leveraging Natural Language Processing (NLP) techniques through the proposed nt2txt approach. By treating payload data as a textual sequence and segmenting it into fixed-size tokens, our method enables the application of sequence modeling to raw traffic data without relying on metadata such as IP addresses or port numbers, which are known to introduce bias. Furthermore, the subdivision of flows into semi-flows, combined with strict dataset partitioning, reduces spurious correlations and improves the generalizability of the model. The experimental results, validated using an LSTM model, demonstrate a clear improvement over similar architectures that do not incorporate bias-reduction measures. Achieving $88{,}87 \pm 0{,}04\%$ accuracy, along with high values in MCC and Cohen's Kappa, confirms the robustness and practical viability of our approach. Importantly, the methodology is device- and protocol-agnostic, scalable to large datasets, and suitable for real-time deployment in environments such as 4G/5G networks, which face growing demands in traffic analysis and cybersecurity. Future research should explore the integration of more advanced deep learning architectures (e.g., Transformers) that could further improve classification performance. Additionally, applying this methodology to other types of encrypted traffic could provide valuable insights into its robustness and adaptability.

Acronyms

AI	Artificial Intelligence
CLI	Command-Line Interface
CNN	Convolutional Neural Network
CV	Cross Validation
DL	Deep Learning
DPI	Deep Packet Inspection
DT	Decision Trees
KNN	K-Nearest Neighbors
LSTM	Long Short-Term Memory
MCC	Matthew's Correlation Coefficient
ML	Machine Learning
MLP	Multi-layer Perceptron
NLP	Natural Language Processing
NT	Network Traffic
NTC	Network Traffic Classification
PCAP	Packet Capture
QoS	Quality of Service

RNN	Recurrent Neural Network
SPI	Surface Packet Inspection
SVM	Support Vector Machine
VoIP	Voice over IP

Acknowledgements
This publication is part of the Project: I+D+i "TECT-CI5G: Tecnologías Transfor-madoras en Ciberseguridad: Aprendizaje Profundo y Minería de Grafos para la Inte-ligencia en Tráfico de Red 5G" Ref. CPP2023-010710 funded by MICIU/AEI/10.13039/501100011033 and FEDER, UE.

Author contributions
AM contributed to the design of the study, conceptualized the software and methodology, validated the analysis, interpreted the data, and drafted the manuscript. CAL implemented the software, curated and provided the data, conducted a literature review, and reviewed the study. EA contributed to the study design, supervised the research, and reviewed the manuscript. AGT also contributed to the study design, supervised the research, and reviewed the manuscript.

Funding
Not applicable.

Data availability
No datasets were generated or analysed during the current study.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare no competing interests.

Received: 1 March 2025 / Accepted: 12 May 2025
Published online: 23 December 2025

References

1. Umair M, Iqbal Z, Bilal M, Nebhen J, Almohamad T, Mehmood R. An efficient internet traffic classification system using deep learning for IoT. Comput Mater Contin. 2021;71(1):407–22.
2. Cheng G, Wang S. Traffic classification based on port connection pattern. En: 2011 International Conference on Computer Science and Service System (CSSS). 2011. pp. 914-7. Web: https://ieeexplore.ieee.org/document/5974374
3. Azab A, Khasawneh M, Alrabaee S, Choo KKR, Sarsour M. Network traffic classification: Techniques, datasets, and challenges. Digit Commun Netw. 18 de septiembre de 2022; Web: https://www.sciencedirect.com/science/article/pii/S2352864822001845
4. Maitin AM, Romero Muñoz JP, García-Tejedor ÁJ. Survey of machine learning techniques in the analysis of EEG signals for Parkinson's disease: A systematic review. Appl Sci Enero De. 2022;12(14):6967.
5. Samuel AL. Some Studies in Machine Learning Using the Game of Checkers. II—Recent Progress. En: Levy DNL, editor. Computer Games I. New York, NY: Springer; 1988. pp. 366–400. Web: https://doi.org/10.1007/978-1-4613-8716-9_15
6. Explaining Deep Learning Models for Per-packet Encrypted Network Traffic Classification. Web: https://ieeexplore.ieee.org/document/9887744/
7. Abbasi M, Shahraki A, Taherkordi A. Deep learning for network traffic monitoring and analysis (NTMA): A survey. Comput Commun 15 De Marzo De. 2021;170:19–41.
8. Lin X, Xiong G, Gou G, Li Z, Shi J, Yu J, ET-BERT:. A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification. En: Proceedings of the ACM Web Conference 2022. New York, NY, USA: Association for Computing Machinery; 2022. pp. 633–42. (WWW '22). Web: https://doi.org/10.1145/3485447.3512217
9. Rombach R, Blattmann A, Lorenz D, Esser P, Ommer B. High-Resolution Image Synthesis with Latent Diffusion Models. arXiv; 2022. Web: http://arxiv.org/abs/2112.10752
10. Senior AW, Evans R, Jumper J, Kirkpatrick J, Sifre L, Green T, et al. Improved protein structure prediction using potentials from deep learning. Nat Enero De. 2020;577(7792):706–10.
11. OpenAI, Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I et al. GPT-4 Technical Report. arXiv; 2023. Web: http://arxiv.org/abs/2303.08774
12. Vinayakumar R, Soman KP, Poornachandran P. Evaluating shallow and deep networks for secure shell (ssh)traffic analysis. En: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). 2017. pp. 266– 74. Web: https://ieeexplore.ieee.org/document/8125851
13. Yang H, He Q, Liu Z, Zhang Q. Malicious encryption traffic detection based on NLP. Secur Commun Netw. 3 de agosto de 2021;2021:e9960822.
14. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN et al. Attention Is All You Need. arXiv; 2017. Web: http://arxiv.org/abs/1706.03762

15. Cheng J, He R, Yuepeng E, Wu Y, You J, Li T. Real-Time Encrypted Traffic Classification via Lightweight Neural Networks. En: GLOBECOM 2020–2020 IEEE Global Communications Conference. 2020. pp. 1–6. Web: https://ieeexplore.ieee.org/document/9322309
16. Seydali M, Khunjush F, Akbari B, Dogani J. CBS: A deep learning approach for encrypted traffic classification with mixed Spatio-Temporal and statistical features. IEEE Access. 2023;11:141674–702.
17. Seydali M, Khunjush F, Dogani J. Streaming traffic classification: a hybrid deep learning and big data approach. Clust Comput 1 De Julio De. 2024;27(4):5165–93.
18. Zhao J, Jing X, Yan Z, Pedrycz W. Network traffic classification for data fusion: A survey. Inf Fusion 1 De Agosto De. 2021;72:22–47.
19. Lopez-Martin M, Carro B, Sanchez-Esguevillas A, Lloret J. Network traffic classifier with convolutional and recurrent neural networks for internet of things. IEEE Access. 2017;5:18042–50.
20. Aceto G, Ciuonzo D, Montieri A, Pescapé A. Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges. IEEE Trans Netw Serv Manag Junio De. 2019;16(2):445–58.
21. Aceto G, Ciuonzo D, Montieri A, Pescapé A. Mobile Encrypted Traffic Classification Using Deep Learning. En: 2018 Network Traffic Measurement and Analysis Conference (TMA). 2018. pp. 1–8. Web: https://ieeexplore.ieee.org/document/8506558
22. Ting KM. Confusion Matrix. En: Sammut C, Webb GI, editores. Encyclopedia of Machine Learning. Boston, MA: Springer US; 2010. pp. 209–209. Web: https://doi.org/10.1007/978-0-387-30164-8_157
23. A Coefficient of Agreement for Nominal Scales - Jacob Cohen. 1960. Web: https://journals.sagepub.com/doi/10.1177/001316446002000104
24. Gorodkin J. Comparing two K-category assignments by a K-category correlation coefficient. Comput Biol Chem 1 De Diciembre De. 2004;28(5):367–74.
25. Pattern Recognition and Machine Learning. Web: https://link.springer.com/book/9780387310732
26. VPN. 2016| Datasets| Research| Canadian Institute for Cybersecurity| UNB. Web: https://www.unb.ca/cic/datasets/vpn.html
27. Aceto G, Ciuonzo D, Montieri A, Pescapè A. MIMETIC: mobile encrypted traffic classification using multimodal deep learning. Comput Netw 24 De Diciembre De. 2019;165:106944.

Publisher's note