

From Packets to Tokens: A Transformer Framework for Real-Time Network Traffic Classification

Arancha Ciprian Encarnación

`arancha.ciprian@ceiec.es`

Universidad Francisco de Vitoria

Ana Maria Maitín

Universidad Francisco de Vitoria

Carlos Arranz

Universidad Francisco de Vitoria

Alberto Nogales

Universidad de Alcalá

Jose Manuel Sanchez Velázquez

Universidad Francisco de Vitoria

Emilio Alba

Foqum Analytics SL.

Research Article

Keywords: Deep Learning, Transformers, Sequence Modelling, Network Traffic Analysis, Real-Time Systems, Cybersecurity

Posted Date: October 23rd, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11064762/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

From Packets to Tokens: A Transformer Framework for Real-Time Network Traffic Classification

Arancha Ciprian Encarnación¹, Ana M. Maitín¹, Carlos Arranz¹, Alberto Nogales², Jose M. Sanchez Velázquez¹, Emilio Alba³

¹ CEIEC, Universidad Francisco de Vitoria, 28223, Madrid, Spain

² University of Alcalá (UAH) Computer Science Department, Alcalá de Henares, Spain

³ Foqum Analytics SL, C. Agustín Durán 24, Madrid, Spain

arancha.ciprian@ceiec.es

Corresponding author: Arancha Ciprian Encarnación

arancha.ciprian@ceiec.es

Abstract. The increasing volume, diversity, and encryption of modern Internet communications have made network traffic classification an essential yet challenging cybersecurity task. Traditional approaches based on ports, payload inspection, or handcrafted features struggle to generalise across heterogeneous environments and encrypted protocols. Inspired by recent advances in Natural Language Processing (NLP), this work frames network traffic analysis as a sequence modelling problem, in which packet payloads are converted into numerical token-based representations. We propose an end-to-end framework that combines a preprocessing pipeline with a Transformer encoder operating directly on token-level network traffic. The approach is evaluated on five classes from the ISCXVPN2016 dataset: Chat, Email, File Transfer, Streaming, and VoIP, achieving 93.11% test accuracy, showing strong overall performance on the test set despite severe class imbalance. Finally, real-time experiments on live network captures support the framework's practical applicability for online network monitoring.

Keywords: Deep Learning; Transformers; Sequence Modelling; Network Traffic Analysis; Real-Time Systems; Cybersecurity

1 Introduction

The rapid growth of Internet services, encrypted communications, and heterogeneous applications has significantly increased the complexity of network traffic analysis. Network traffic classification has consequently become a fundamental component of modern cybersecurity infrastructures, enabling intrusion detection, anomaly identification, quality-of-service management, resource allocation, and network monitoring. However, the widespread adoption of encryption mechanisms, the continuous emergence of new services, and the massive volume of exchanged data have reduced the effectiveness of traditional approaches based on port inspection, protocol signatures, and handcrafted features. These challenges demand intelligent

solutions that can automatically learn robust, generalisable representations directly from raw network traffic. This is the case of techniques such as Artificial Intelligence (AI), which has demonstrated a high ability to manage large amounts of complex data.

AI is a branch of computer science that explores the mechanisms behind human intelligence and behaviour by replicating these behaviours in machines, not necessarily through the same processes (Russell and Norvig 2016). Among all the AI techniques, Deep Learning (DL) models have stood out from the rest in recent years. They were introduced by (LeCun, Bengio, and Hinton 2015) as models with a hierarchical, multilayered structure capable of learning data representations at progressively higher levels of abstraction, starting from the raw input.

The motivation of this work is not only to develop a DL traffic analysis model but also to leverage methods from Natural Language Processing (NLP) and adapt them to the paper’s purpose. This decision assumes that both texts and network traffic exhibit a hierarchical and sequential structure, making their analysis fundamentally similar. In NLP, sentences are composed of words and characters, while in network traffic, packets form flows that belong to broader communication sessions. Contextual dependency is crucial in both domains; just as the meaning of a word depends on its surrounding words, the behaviour of network traffic is influenced by previous packets in a session.

The novelty of this work lies in providing a complete pipeline that captures network traffic in real time, preprocesses packet payloads into token-based semi-flow representations and classifies the resulting traffic using a Transformer-based model. The contribution of this work is not the introduction of Transformers, textual traffic representations, or real-time inference in isolation. Instead, we integrate them into a unified end-to-end framework that covers the process from network traffic capture to real-time classification. The proposed methodology transforms captured traffic into packet sequences represented through payload tokens, models contextual dependencies among packets, and produces a final classification using a Transformer-based encoder without recurrent components.

Results show that, when evaluated on five communication categories, namely Chat, Email, File Transfer, Streaming, and VoIP, the proposed framework achieves strong in-distribution performance, with a test accuracy of 93.11%, a Cohen's Kappa score of 0.8954, and a Matthews Correlation Coefficient (MCC) of 0.901, despite the pronounced class imbalance present in the dataset. Real-time experiments on live traffic also show that the system can process network captures fast enough for online classification, although performance varies across traffic classes. The paper is organised as follows. Section 2 reviews relevant work on analysing network traffic using different techniques. Section 3 formally describes the dataset used for training, along with the DL models employed in the study. Section 4 presents the obtained results and their interpretation. Finally, Section 5 discusses the conclusions and outlines directions for future research.

2 Related work

As stated above, this work aims to implement a Transformer model to classify network traffic. In this way, the field of cybersecurity has experienced a notable development from classical approaches to newer ones based on DL techniques. The following paragraphs summarise representative works from the state of the art.

Among the most classic network traffic classifiers, we can highlight techniques such as port inspection or Deep Packet Inspection (DPI). (Guang Cheng and Song Wang 2011) is framed in the first group and analyses server connection patterns and simultaneous connections across different services, proposing a traffic classification method based on port inspection. In the second group, we can highlight (Hubballi and Khandait 2022), which proposes an efficient DPI-based network flow classification method using application keywords, their order, and heuristics to improve speed. Rule-based classifiers, such as those used by (Duffield et al. 2009), detect traffic anomalies using IP flow information and predefined rules. Finally, (Sinadskiy and Domukhovskiy 2020) integrates entropy and statistical algorithms to develop a method that identifies network nodes and key fields in traffic with unknown protocols. The main difference between these works and the present one is that none of them uses a DL model such as Transformers.

Other works benefit from the application of classical ML models. Papers like (Jonathan, Misra, and Osamor 2021) investigate the effectiveness of feature selection in ML-based network traffic classification, using a dataset with various applications like Secure Copy Protocol (SCP) and Secure File Transfer Protocol (SFTP), and comparing ML models such as SVM, Naïve Bayes, k-NN, and Decision Trees. In (Koumar and Cejka 2022), the paper analyses encrypted network traffic using periodic behaviour detection, leveraging time series analysis with autocorrelation and Lomb-Scargle periodograms. The authors trained an XGBoost classifier, achieving a 90% F1-score, to categorise 61 types of network traffic, including social networks, antivirus software, and multimedia applications, aiding malicious traffic detection. Also, (Kumar et al. 2021) focused on encrypted network traffic classification; this study applied four ML models to an encrypted traffic dataset containing 75 applications, although the reported results focus on 11 application classes. The Decision Tree model achieved 99.8% accuracy, making it highly effective for network monitoring, ISP (Internet Service Provider) management, and cybersecurity applications. Another interesting work is (Menezes and Mello 2021), which explores flow feature-based classification as an alternative to port- and payload-based methods, which struggle with encryption and obfuscation. By using KNN, Decision Trees, and DBSCAN, the study shows that supervised learning effectively classifies network flows for tasks like performance monitoring and security anomaly detection. Finally, (AHMED et al. 2023) integrates Artificial Neural Networks (ANN) and SVM to detect and mitigate DDoS (Distributed Denial of Service) and packet suppression attacks. If we compare these papers with the present one, we can see that none of them uses NLP-based Deep Learning models, and in some cases, although the work classified network traffic, it was not applied to the field of cybersecurity.

In the case of DL models, and specifically Transformers, applied to network traffic analysis, several relevant works can be highlighted. (Izadi, Ahmadi, and Rajabzadeh 2022) integrates Deep Belief Networks, CNN, and MLP with Bayesian decision fusion, achieving 97% accuracy on ISCXVPN2016 but classifying between VPN and non-VPN. (Zhang et al. 2023) proposes a hybrid DL framework that takes the previous spatio-temporal mobile network traffic maps as input and combines temporal convolutional networks with a sparse-attention Transformer for predicting mobile network traffic. (Liu et al. 2025) presents TransECA-Net, a particular type of CNN and Transformer encoders that enhances encrypted traffic classification for the ISCX VPN-nonVPN dataset. Beyond traffic classification, Transformers have been increasingly adopted for network intrusion detection: (Manocchio et al. 2024) introduce FlowTransformer, a framework applying Transformers to flow-based intrusion detection, discriminating between benign and malicious traffic. (Nguyen and Watabe 2022) apply BERT-based language modelling to network flow sequences, leveraging NLP representations for intrusion detection (benign vs malicious). Finally, (Domi et al. 2025) propose NetPacketformer, a real-time context-aware packet-level Conformer-Transformer for intrusion detection that shares conceptual similarities with our approach. Unlike NetPacketformer, which applies a Conformer to fixed-length sequences of normalised raw packets for real-time intrusion detection, our framework performs application traffic classification by transforming captured packet payloads into token-based semi-flow representations and processing them with a Transformer-based encoder without recurrent components.

Summarising, none of the previous works simultaneously satisfies the following conditions: (1) a Transformer-based encoder without recurrent components; (2) an end-to-end pipeline that converts captured packet payloads into token-based semi-flow representations; (3) demonstrated real-time applicability on live captured traffic; and (4) a systematic cross-application evaluation.

3 Materials and methods

This section describes the dataset used to train the models, the Deep Learning methods applied, and the metrics used to evaluate model performance.

3.1 A dataset of real network traffic

After reviewing the state of the art, several public datasets suitable for this research were identified. Among them, ISCXVPN2016 stands out due to its proven validity and rigour, as recognised in numerous studies (Xian 2021). This dataset includes network traffic captures in PCAP format covering a variety of communication scenarios. In Table 1, a summary of how these classes are distributed can be found.

Table 1. Features of the 5 selected classes from ISCXVPN2016

Categories	Number of applications	Number of PCAP files	Number of packets	Recording time (min)	Percentage of preprocessed semi-flows
------------	------------------------	----------------------	-------------------	----------------------	---------------------------------------

Chat	6	20	153,451	164.12	0.19 %
File transfer	4	32	11,007,165	190.88	38.65 %
Streaming	7	30	4,991,536	294.65	30.07 %
VoIP	4	23	5,682,877	726.23	31.06 %
Email	1	4	75,079	207.45	0.02 %
Total	22	109	21,910,108	1,583.33	100 %

3.2 Transformers

In this work, we employed Transformers, a groundbreaking architecture introduced by (Vaswani et al. 2017). Transformers follow an encoder-decoder structure, where the encoder processes input data into a compact representation containing essential information, and the decoder uses the encoder output to generate the final output sequence. Both components consist of stacks formed by identical blocks. Each encoder block primarily comprises a multi-head self-attention mechanism and a position-wise Feed-Forward Network (FFN). To facilitate deeper architectures, residual connections described by (He et al. 2016) are applied around each module, followed by Layer Normalisation (Ba, Kiros, and Hinton 2016). The original Transformer encoder consists of six identical layers, each with two sublayers: a self-attention mechanism and a feed-forward network. The encoder's role is to encode a given input token while considering the context provided by other relevant inputs. The decoder follows a similar structure but includes an additional multi-head attention sublayer, allowing it to interact with the encoder's output for improved information retrieval and generation.

In our work, we use only the encoder component of the Transformer, since the task is classification rather than sequence generation. The input to the encoder is a semi-flow representation, where packets are represented through payload-derived tokens.

3.3 Evaluation metrics

Since we are dealing with a multiclass classification problem, we first use accuracy as a guiding metric to assess the initial performance of the models. Accuracy is computed as the number of correctly classified samples divided by the total number of classified samples, as it is described in Equation 1.

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

Regarding the equation above, TP (True Positives) are instances correctly classified as the expected class. TN (True Negatives) are instances correctly classified as not belonging to the class under consideration. Finally, N represents the total number of classified instances.

This metric is good for measuring the initial performance of the model, but we should add other metrics considering False Positives (FP) and False Negatives (FN). Equations 2 and 3 formalise the Sensitivity and Specificity for each class using a one-vs-rest interpretation.

$$Sensitivity = \frac{TP}{TP+FN} \quad (2)$$

$$Specificity = \frac{TN}{TN+FP} \quad (3)$$

By examining the previous equation, FN refers to instances that belong to a specific class but are incorrectly classified as one of the other classes. FP refers to instances that belong to a different class but are incorrectly classified as the given class.

The interpretation of these metrics in multiclass problems can be limited, so additional metrics are introduced to better assess model performance.

Another interesting metric is Matthews Correlation Coefficient (MCC). This is defined by (Gorodkin 2004) as the correlation between actual and predicted labels in a classification task involving N items distributed across K categories. Equation 4 formalises this metric.

$$MCC = \frac{Nc - \sum_{k=1}^K t_k p_k}{\sqrt{N^2 - \sum_{k=1}^K t_k^2} \sqrt{N^2 - \sum_{k=1}^K p_k^2}}, \quad (4)$$

If we analyse the previous Equation, c denotes the number of correctly classified samples, t_k represents the actual number of samples in category k , and p_k corresponds to the number of samples predicted in category k . The MCC ranges from $[-1, 1]$, where 1 indicates perfect classification, 0 signifies performance equivalent to random guessing, and -1 represents total disagreement between predictions and actual observations.

Finally, Cohen's kappa score quantifies the agreement between two evaluators in a classification task where N items are assigned to K mutually exclusive categories, (Cohen 1960). This metric is defined in Equation 5.

$$\kappa = \frac{p_0 - p_e}{1 - p_e}, \quad (5)$$

The previous Equation can be understood as follows. p_0 is the agreement rate between assessors; that is, $p_0 = (NumberHits)/N$, and p_e is the probability of agreement if the assessors classify the samples randomly as described in Equation 6.

$$p_e = \sum_{k=1}^K \frac{n_{1k}}{N} \frac{n_{2k}}{N} \quad (6)$$

In the Equation above, several terms are defined. n_{ik} represents the number of items classified into a category k by assessor i . When there is complete agreement, $\kappa = 1$. If their agreement is no greater than would be expected by chance, then $\kappa = 0$. Notably, κ can take negative values, indicating a lack of correlation between the assessors' classifications, which may result from random chance or an underlying systematic disagreement.

4 Results

In this section, we describe the developed workflow, which consists of preprocessing the raw traffic and training a Transformer-based classifier. First, raw PCAP files are transformed into semi-flow representations based on payload tokens, so the data can be fed into the model. This preprocessing stage is carried out in two steps: first, the creation of semi-flows, and second, a transformation called nt2txt (Maitín et al. 2025), which converts packet payloads into a numerical token-based format. Second, we train a Transformer-based model for network traffic classification, applying the best practices described by (Alberto Nogales, Ana M. Maitín, and Álvaro J. García-Tejedor 2026). Fig. 1 shows the workflow applied in this research.

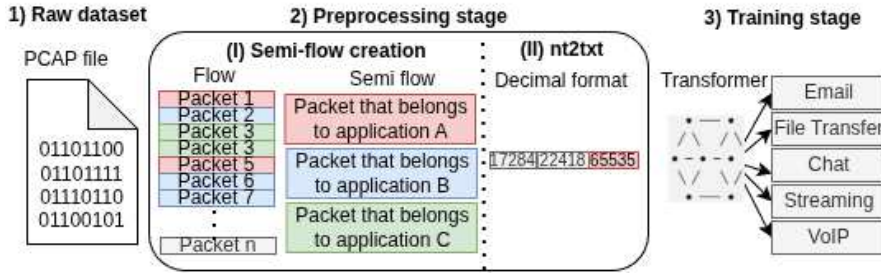


Figure 1. Workflow to train a Transformer-based network traffic classifier.

4.1 From PCAP to tokens: a preprocessing method

As described above, the dataset presents issues related to data volume and class imbalance. Additionally, the nature of network traffic may introduce bias in the training dataset, since packets from different flows can be sequentially mixed, resulting in an inaccurate representation of the temporal sequence of events. To avoid this, we apply the two-step preprocessing method nt2txt, which takes packet-level traffic data as input, structures packets into flows and semi-flows, and converts packet payloads into numerical tokens that can be fed into DL models. In the first step, packets that are not useful for the model are discarded, and flows are segmented into semi-flows, which are subsets of the same flow containing a fixed number of packets. During the second step, called nt2txt, the semi-flows are converted into numerical token vectors, so they can be introduced into DL models. The output of the preprocessing stage provides a structured representation of network traces, preserving the sequential nature of events to facilitate feature extraction and the identification of relationships among packets. Due to this preprocessing method, different feature values must be chosen depending on the dataset. In the first step, the size of each semi-flow is determined by two hyperparameters: the maximum number of packets it can contain and the maximum duration, defined as the time difference between the first and last packet. Also, the minimum number of packets in a flow must be decided. During the second step, two values must be selected: the payload length of a packet and the number of charac-

ters per token. These values define a search space where different combinations are tested during model training.

Creating the semi-flows. The first step of the preprocessing method consists of building flows from the captured traffic before segmenting them into semi-flows. First, packets without payload are removed, since they do not provide useful input to the model. Then, the remaining packets are grouped into flows using selected packet fields, such as source and destination addresses, protocol, and ports. This grouping is necessary because packets from different applications may appear mixed in real traffic, as illustrated in Fig. 2. Once packets have been assigned to their corresponding flows, each flow is split into semi-flows with a fixed number of packets and a maximum duration. Fig. 2 describes the process followed in this step.

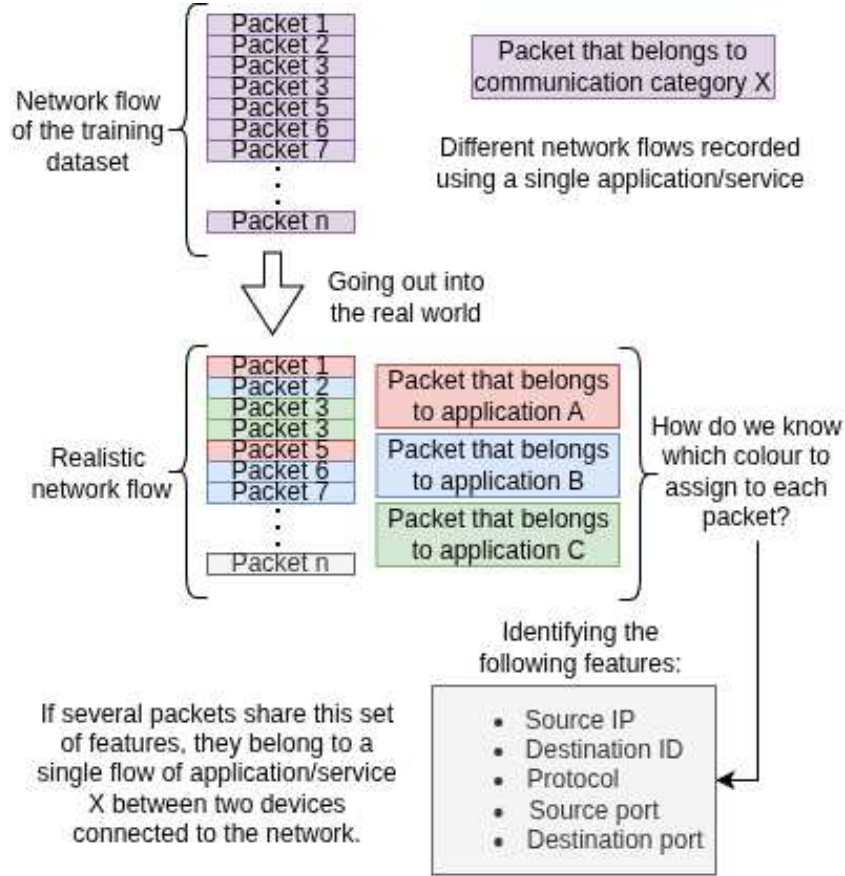


Figure 2. Process to obtain flows and semi-flows.

Nt2txt. In the second step of the pre-processing stage, each semi-flow is converted into a numerical representation that can be used as input to the Transformer model. A

semi-flow is composed of several packets, and each packet contains a payload. For each packet in the semi-flow, the payload is cleaned and split into hexadecimal tokens of equal size. Each hexadecimal token is then converted into decimal format, and an additional end-of-packet token is added to delimit the packet. As a result, each packet is represented as a vector of decimal tokens, and each semi-flow is represented as a sequence of packet vectors. Fig. 3 shows how packet payloads are converted into decimal token vectors.

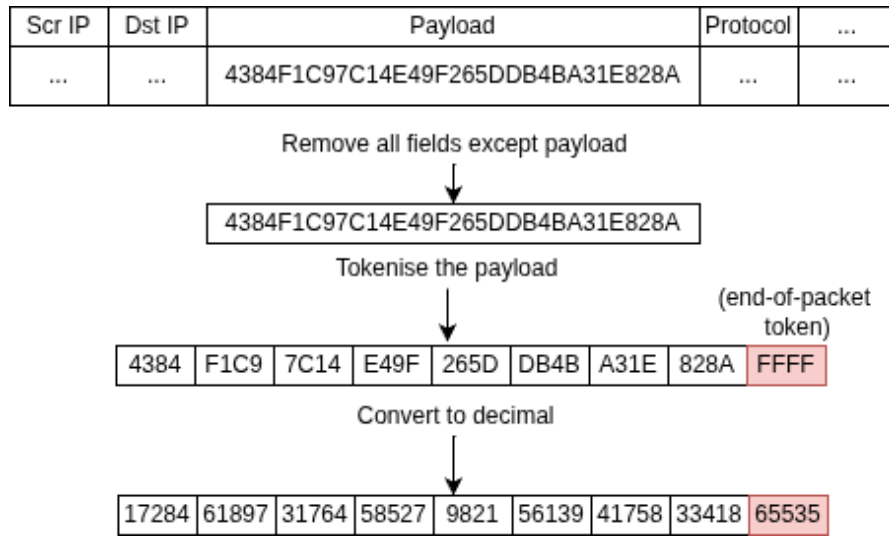


Figure 3. The nt2txt process for converting payloads.

Finding the optimal features for preprocessing the data. As mentioned above, the applied preprocessing method requires several feature values to be selected. To obtain the best set of values, different combinations were tested as part of the pipeline used to train the Transformer. This information is compiled in Table 2.

Table 2. Search space of features for dataset preprocessing

Dataset feature	Tested values
Minimum packets per flow	20, 100, 150
Number of packets per semi-flow	5, 10, 20, 50, 100, 150
Maximum duration of a semi-flow (seconds)	15, 30, 45
Payload length of a packet	200, 400, 600, 800
Number of characters per token	4, 8

The best preprocessing values were selected based on the Transformer model's performance. In this case, these values are as follows. The minimum number of packets per flow is set to 20. The number of packets per semi-flow is also 20. The maximum

duration of a semi-flow is 30 seconds. Additionally, the payload length of a packet is 800, and the number of characters per token is 4. Using these values in the pre-processing process, the pre-processed dataset is distributed as follows. Chat has 1,325 semi-flows, Email 168, File Transfer 266,577, Streaming 207,405 and VoIP 214,180. This makes a total of 689,655 semi-flows. Each semi-flow contains 20 packets, and each packet is represented by 201 tokens. It should be highlighted that the class imbalance shown in the pre-processed dataset is a problem that other works have faced by applying sampling strategies, but in this case, we were able to directly train an accurate model.

4.2 A Transformer-based network traffic classifier

To obtain a good classifier for network traffic, we applied the following steps: Creation of the training, validation, and test subsets. Fine-tuning of the model’s hyperparameters. Interpretation of the metrics. Each of these steps is described in depth below.

Train/validation/test subsets. The dataset was first divided into a development subset (90%) and an independent test subset (10%). The test subset was kept separate and used exclusively for the final evaluation of the trained model. Five-fold cross-validation was then applied to the 90% development subset, which was iteratively divided into training and validation folds during model development. The reported training and validation metrics therefore correspond to the mean performance across the five folds, together with their standard deviations. This procedure provides a more robust estimate of model performance in the presence of the pronounced class imbalance of the dataset.

Hyperparameter fine-tuning. To identify the best-performing hyperparameter combination, a grid search approach was implemented. Grid search systematically explores different hyperparameter values and selects the combination that yields the best performance (Bergstra et al. 2011). Table 3 compiles all the hyperparameters that were evaluated and their values.

Table 3. Hyperparameter tuning for the Transformer model.

Hyperparameter	Tested values
Batch size	100, 500, 1,000, 2,500
Epochs	5, 15, 30, 50
Learning rate	3e-4
Number of attention layers	1
Positional encoding	No
Number of neurons	250
Dropout	0.20, 0.50
Number of attention heads	4, 8

Metrics performance. At this point, we have trained a Transformer model with the aim of classifying network traffic. The first way to measure the performance of the model is by using the accuracy metric as a guide. We have obtained the value of this metric at the training, validation, and test stages. For training and validation, the average values and their standard deviations were computed across the five folds, whereas test accuracy was obtained on the independent test subset.

The five-class Transformer model achieved training, validation, and test accuracies of $93.32\% \pm 0.04\%$, $92.79\% \pm 0.36\%$, and 93.11% , respectively. The small differences among these values indicate stable learning dynamics and suggest that the model generalises well to unseen in-distribution data without exhibiting significant overfitting. Complementary agreement-based metrics further support this conclusion, with a Cohen's Kappa score of 0.89 and an MCC of 0.90. The former denotes an almost perfect agreement between predicted and true labels beyond chance, while the latter reflects excellent overall predictive performance by accounting for all classes and remaining robust in the presence of class imbalance. Collectively, these results suggest that the learned representations capture genuine and discriminative traffic characteristics, highlighting the strong in-distribution performance of the proposed five-class traffic classification model.

Next, it is essential to conduct a detailed analysis of the model's performance considering FN and FP. In the context of network traffic classification, FP and FN can have significant implications as stated above. Therefore, minimising both types of errors is crucial for reliable network traffic classification. On the test set, the model obtained a weighted-average sensitivity specificity, precision, and F1-score values of 93.11% , 95.83% , 94.03% , and 92.97% , respectively, indicating a strong balance between correctly identifying traffic classes and minimising misclassifications. The high sensitivity demonstrates that most network semi-flows are assigned to their correct categories, while the elevated specificity confirms the model's ability to avoid incorrectly labelling flows as belonging to a particular class. From an operational perspective, these properties are essential for traffic management systems, as they reduce the risk of inappropriate quality-of-service policies, inefficient resource allocation, or erroneous security decisions caused by misclassified traffic.

To complement the global performance metrics, Table 4 reports the sensitivity and specificity achieved for each traffic category. These metrics are particularly relevant in network traffic analysis, as high sensitivity indicates that semi-flows belonging to a given traffic class are correctly identified, whereas high specificity minimises false alarms and prevents incorrect traffic labelling that could adversely affect quality-of-service policies, bandwidth management, or security mechanisms. Apart from that, Figure 4 depicts the confusion matrix for the five-class network traffic classification problem. The matrix provides a fine-grained evaluation of the model by revealing how each traffic category is assigned and which classes are most susceptible to misclassification. Such information is essential for understanding the practical limitations of the classifier and for identifying traffic types that share overlapping behavioural patterns.

Table 4. Per-class performance of the five-class Transformer model.

Class	Sensitivity	Specificity
Chat	73.48%	99.82%
Email	5.88%	100.00%
File Transfer	99.74%	89.58%
Streaming	78.05%	99.74%
VoIP	99.63%	99.81%

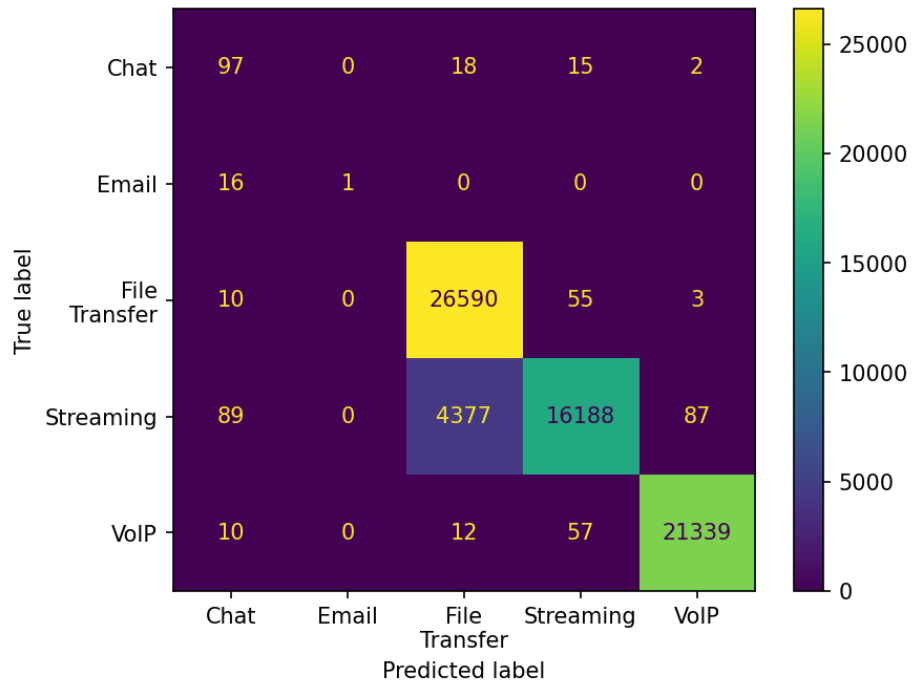


Figure 4. Confusion matrix for the five-class network traffic classification.

Based on the previous Figure and considering the other metrics defined above, the proposed Transformer exhibits heterogeneous class-wise performance, with particularly strong results for File Transfer and VoIP traffic, both achieving sensitivities above 99%. The principal classification challenge arises from Streaming traffic being misclassified as File Transfer, which reduces the sensitivity of the Streaming class to 78.05%. Similarly, Chat traffic presents moderate confusion with these categories. Although Email attains perfect specificity, its low sensitivity may be related to the severe class imbalance and the limited number of available samples. Overall, the results confirm the effectiveness of the proposed approach while emphasising classification difficulties associated with inter-class similarities and the pronounced data imbalance affecting minority categories.

Description of the best model. At this point, we have been able to obtain a Transformer model that performs well. The best model takes as input semi-flows of shape (20, 201), where 20 is the number of packets per semi-flow, and 201 is the number of tokens per packet. The architecture consists of a single Transformer encoder block followed by classification layers. The encoder block contains a MultiHeadAttention layer with 8 heads and a head size of 6, implementing 38,937 trainable parameters. A residual connection is applied around the attention output, followed by Layer Normalisation (402 parameters). The feed-forward network consists of two one-dimensional convolutional layers. The first layer expands the representation to 256 filters, comprising 51,712 trainable parameters, while the second layer projects the features back to a 201-dimensional embedding space, adding 51,657 trainable parameters. No positional encoding is applied in the selected configuration. After the encoder block, a GlobalAveragePooling1D layer reduces the output to a vector of 20 features, followed by a dense layer with 250 neurons and a final output layer with 5 neurons (one per traffic class). Dropout with a rate of 0.2 is configured in both the Transformer encoder and MLP. The total number of trainable parameters is 149,213. The best-performing hyperparameters, identified through the grid search described above, are: 15 epochs, a batch size of 100 and a learning rate of 3×10^{-4} . The model was trained on an NVIDIA GeForce RTX 3090 GPU (24 GB VRAM).

Controlled Per-Class Real-Time Evaluation. To rigorously assess the real-time applicability of the proposed framework, a controlled per-class evaluation was conducted on live network traffic. TShark was executed on a Wi-Fi network interface, generating a new PCAP file every 8 seconds. Each PCAP was immediately processed through the nt2txt preprocessing pipeline and classified by the Transformer model. For each of the five traffic classes, three independent experimental runs were performed. Each run consisted of ten consecutive captures, and each capture corresponded to an 8-second traffic window, yielding 30 capture windows per class. Each valid capture window was then processed to obtain one window-level classification decision. Windows that did not produce a valid classification decision were excluded from the analysis, resulting in 30 decisions for Chat, Streaming, and VoIP, 29 for File Transfer, and 20 for Email. Prior to each run, all processes were terminated and a 15-second stabilisation period was observed to minimise interference from extraneous traffic. For each valid capture window, the Transformer classified the generated semi-flows, and the dominant predicted class was used as the window-level classification.

For each traffic class, the application used during live capture was selected to approximate the traffic characteristics represented in the ISCXVPN2016 training set. Chat traffic was generated using Gmail Chat, the direct successor of Google Hangouts, which is explicitly represented in the training corpus. Streaming traffic was produced with Spotify as the reference streaming application. For VoIP, a synthetic G.711 RTP stream (a standard format for transmitting voice over IP networks) was generated to simulate the type of voice communication included in the VoIP class of the dataset. File Transfer traffic was captured during sustained HTTPS file downloads, with the aim of reproducing a sustained bulk transfer pattern. Finally, Email traffic was cap-

tured while sending and receiving emails using the Thunderbird email client, matching the email traffic included in the original dataset.

Table 5 summarises the per-class real-time classification accuracy, mean latency decomposed into preprocessing (Pre) and inference (Inf) stages, and the Real-Time Factor (RTF), defined as the ratio of the measured preprocessing and inference time to the capture window duration (8 seconds). An RTF below 1.0 indicates that the measured processing time is shorter than the 8-second capture window, supporting real-time capability.

Table 5. Per-class real-time classification results.

Class	Windows	Accuracy	95% CI	Pre (ms)	Inf (ms)	Total (ms)	RTF
Chat	30	93.3%	[78.7-98.2%]	65.6	38.0	103.6	0.013
Streaming	30	96.7%	[83.3-99.4%]	103.1	37.3	140.5	0.018
VoIP	30	100.0%	[88.6-100.0%]	41.2	29.5	70.7	0.009
File Transfer	29	100.0%	[88.3-100.0%]	1546.2	543.1	2089.3	0.261
Email	20	0.0%	[0.0-16.1%]	45.2	10.8	56.0	0.007

Across all classes, the RTF remains below 1.0, indicating that the measured preprocessing and inference stages operate faster than the 8-second capture window. The preprocessing stage consistently dominates total latency, accounting for approximately 58-81% of processing time across Chat, Streaming, VoIP, and Email, with relatively short processing times for all four classes. The exception is File Transfer, where both preprocessing (1546 ms) and inference (543 ms) are substantially higher due to the large volume of semi-flows generated by high-throughput HTTPS transfers. Nevertheless, even in this extreme case, the RTF of 0.261 confirms real-time capability. In terms of performance, Chat, Streaming, VoIP, and File Transfer achieved accuracies above 90%, with VoIP and File Transfer reaching 100%. However, the Email class achieved 0% accuracy across all valid capture windows.

Cross-Application Generalisation Analysis. A fundamental question for any traffic classifier is whether its learned representations capture generalisable, class-level features or merely application-specific artefacts tied to the software used during training. A robust classifier should correctly identify Chat traffic regardless of whether it is generated by Gmail Chat or WhatsApp, and Streaming traffic regardless of whether the source is Spotify or Netflix. To evaluate this cross-application robustness, we conducted a systematic out-of-distribution (OOD) analysis in which, for each traffic class, we evaluated the model on newly captured traffic from a representative application not included in the ISCXVPN2016 training subset used to train the model, under identical experimental conditions. This constitutes a direct test of whether the model generalises beyond its training distribution or overfits to the patterns observed during training.

The OOD applications were selected to test the model with traffic generated by applications different from those used in the corresponding reference experiments. Spe-

cifically, WhatsApp Web was used for Chat, Netflix for Streaming, Google Meet for VoIP and BitTorrent for the File Transfer test. Although BitTorrent traffic was not included in the five-class training subset used in this work, it is categorised as P2P rather than File Transfer in the original ISCXVPN2016 dataset. Email was not tested due to its low performance in the previous experiments. Table 6 presents the accuracy obtained for each reference application and its corresponding OOD application.

Table 6. Cross-application generalisation results.

Class	Reference application	Reference accuracy	OOD application	OOD accuracy	Predicted classes
Chat	Gmail Chat	93.3%	WhatsApp Web	35.7%	1.- Chat (36%) 2.- Streaming (36%) 3.- File Transfer (29%)
Streaming	Spotify	96.7%	Netflix	23.1%	1.- File Transfer (62%) 2.- Streaming (23%) 3.- Chat (15%)
VoIP	G.711 RTP stream	100.0%	Google Meet	0.0%	1.- Streaming (100%)
File Transfer	Direct File Download	100.0%	BitTorrent	7.9%	1.- Streaming (47%) 2.- Chat (45%) 3.- File Transfer (8%)

The results reveal class-specific failure modes when the model is evaluated on different applications and communication technologies. VoIP exhibits the most severe degradation: the classifier achieves 0% accuracy on Google Meet, classifying all evaluated windows as Streaming. Google Meet is built on WebRTC, which differs substantially from the G.711 RTP traffic used in the reference real-time experiment. According to (Mahmoud and Abozariba 2024), WebRTC employs modern codecs, encrypted media transport, and congestion-control mechanisms, resulting in traffic patterns that differ markedly from those of constant-bitrate G.711 RTP streams.

Baseline comparison. The primary baseline for comparison is the LSTM-based model reported in Maitin et al. (2025), which used the same nt2txt preprocessing approach and the same five-class ISCXVPN2016 dataset. That work achieved 88.87% accuracy, a Cohen's Kappa of 0.83, and an MCC of 0.84. Our Transformer model achieves 93.11% accuracy, a Cohen's Kappa of 0.89, and an MCC of 0.90 on the same task, representing improvements of 4.24 percentage points in accuracy, 0.06 in Kappa, and 0.06 in MCC, respectively.

5 Conclusions and future work

The aim of this work was to present a complete Transformer-based pipeline for real-time network traffic classification directly from raw PCAP files. The results demonstrate that the nt2txt preprocessing method can be effectively integrated with a Transformer-based architecture for network traffic classification. The proposed model

achieves high overall classification performance, although important differences are observed between traffic classes, particularly for Email. The low number of Email samples in the training data may partly explain this poor result. In addition, several real-time experiments were conducted, demonstrating the feasibility of the proposed pipeline under controlled live-traffic conditions. Finally, the pipeline was evaluated on new traffic data generated by applications and traffic sources not used to train the model, while maintaining the same experimental conditions. The resulting performance degradation highlights the limited cross-application generalisation observed in these experiments and underscores the need for updated benchmark datasets that include modern applications, protocols, and traffic behaviours.

As future work, this pipeline could be extended towards a multi-stage network traffic analysis framework. The real-time classifier developed in this work could first identify the type of network traffic, such as Chat, Streaming, VoIP, or File Transfer. The classified traffic could then be represented using graph-based models to capture the relationships and temporal evolution of network communications. This representation could support subsequent analysis stages aimed at distinguishing normal from malicious behaviour and, when applicable, identifying the corresponding type of cyberattack. In this way, real-time traffic classification would serve as the first stage of a broader cybersecurity analysis framework.

A further research direction is to improve the generalisation of the system to applications and protocols not represented during training. This could be addressed by incorporating updated real-world traffic and by studying graph-based and temporal representations that capture how network behaviour evolves. Beyond traffic characterisation, these representations could also be used to predict future states or changes in network traffic, providing additional information for subsequent classification and cybersecurity analysis.

Funding

This publication is part of the R&D&I project "TECT-CI5G: Tecnologías Transformadoras en Ciberseguridad: Aprendizaje Profundo y Minería de Grafos para la Inteligencia en Tráfico de Red 5G" Ref. CPP2023-010710, funded by MICIU/AEI/10.13039/501100011033 and by ERDF, EU.

Acknowledgements

We would also like to express our deepest gratitude to the late Álvaro J. García-Tejedor, whose contribution to this project was invaluable and irreplaceable. He was the driving force behind this initiative and played a central role in its conceptualisation and methodological design from its very inception. His vision, intellectual leadership, and unwavering commitment laid the foundations upon which this work was built. Moreover, his leadership as Director of CEIEC was instrumental in facilitating the approval and subsequent development of this project. We are profoundly grateful

for his dedication, guidance, and lasting contribution, and we hope that this work serves as a fitting tribute to his legacy.

References

- AHMED, Adnan, Muhammad AWAIS, Mohammad SIRAJ, and Muhammad UMAR. 2023. "Enhancing Cybersecurity with Trust-Based Machine Learning: A Defense against DDoS and Packet Suppression Attacks." *The Eurasia Proceedings of Science Technology Engineering and Mathematics* 23: 262–68. doi:10.55549/epstem.1368266.
- Alberto Nogales, Ana M. Maitín, and Álvaro J. García-Tejedor. 2026. *Best Practices to Train Accurate Deep Learning Models: A General Methodology*.
- Ba, Jimmy Lei, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. *Layer Normalization*. <https://arxiv.org/abs/1607.06450> (September 16, 2026).
- Bergstra, James, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. "Algorithms for Hyper-Parameter Optimization." *Advances in Neural Information Processing Systems* 24: 2546–54.
- Cohen, Jacob. 1960. "A Coefficient of Agreement for Nominal Scales." *Educational and Psychological Measurement* 20(1): 37–46. doi:10.1177/001316446002000104.
- Domi, Armando, Christos Zonios, Giorgos Tatsis, Anastasios Drosou, and Dimitrios Tzovaras. 2025. "NetPacketformer: Real-Time, Context-Aware Network Intrusion Detection with Transformers." In *2025 IEEE International Conference on Cyber Security and Resilience (CSR)*, IEEE, 687–92. doi:10.1109/CSR64739.2025.11130136.
- Duffield, N., P. Haffner, B. Krishnamurthy, and H. Ringberg. 2009. "Rule-Based Anomaly Detection on IP Flows." In *IEEE INFOCOM 2009*, IEEE, 424–32. doi:10.1109/INFOCOM.2009.5061947.
- Gorodkin, J. 2004. "Comparing Two K-Category Assignments by a K-Category Correlation Coefficient." *Computational Biology and Chemistry* 28(5–6): 367–74. doi:10.1016/j.compbiolchem.2004.09.006.
- Guang Cheng, and Song Wang. 2011. "Traffic Classification Based on Port Connection Pattern." In *2011 International Conference on Computer Science and Service System (CSSS)*, IEEE, 914–17. doi:10.1109/CSSS.2011.5974374.
- He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. "Deep Residual Learning for Image Recognition." In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, 770–78. doi:10.1109/CVPR.2016.90.
- Hubballi, Neminath, and Pratibha Khandait. 2022. "Efficient Keyword Matching for Network Traffic Classification." *Computer Communications* 185: 79–91. doi:10.1016/j.comcom.2021.12.021.
- Izadi, Saadat, Mahmood Ahmadi, and Amir Rajabzadeh. 2022. "Network Traffic Classification Using Deep Learning Networks and Bayesian Data Fusion." *Journal of Network and Systems Management* 30(2): 25. doi:10.1007/s10922-021-09639-z.
- Jonathan, Oluranti, Sanjay Misra, and Victor Osamor. 2021. "Comparative Analysis of Machine Learning Techniques for Network Traffic Classification." *IOP Conference Series: Earth and Environmental Science* 655(1): 012025. doi:10.1088/1755-1315/655/1/012025.

- Koumar, Josef, and Tomas Cejka. 2022. "Network Traffic Classification Based on Periodic Behavior Detection." In *Proceedings of the 18th International Conference on Network and Service Management*, International Federation for Information Processing, 359–63. doi:10.23919/cnsm55787.2022.9964617.
- Kumar, S.R., Ch. Vinay, A.M.S. Lakshmi, K. Jahnavi, and S. Karthik. 2013. "Network Traffic Classification on Encrypted Data Using Machine Learning Approach." *International Journal of Innovative Research in Computer and Communication Engineering*.
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. 2015. "Deep Learning." *nature* 521(7553): 436–44. doi:10.1038/nature14539.
- Liu, Ziao, Yuanyuan Xie, Yanyan Luo, Yuxin Wang, and Xiangmin Ji. 2025. "TransECA-Net: A Transformer-Based Model for Encrypted Traffic Classification." *Applied Sciences* 15(6): 2977. doi:10.3390/app15062977.
- Mahmoud, Haitham, and Raouf Abozariba. 2024. "A Systematic Review on WebRTC for Potential Applications and Challenges beyond Audio Video Streaming." *Multimedia Tools and Applications* 84(6): 2909–46. doi:10.1007/s11042-024-20448-9.
- Maitin, Ana M., Carlos Arranz-Luque, Emilio Alba, and Álvaro J. García-Tejedor. 2025. "Application of Natural Language Processing Techniques to Network Traffic Processing for Classification Using Deep Learning Models." *Journal of Big Data* 12(1): 277. doi:10.1186/s40537-025-01183-w.
- Manocchio, Liam Daly, Siamak Layeghy, Wai Weng Lo, Gayan K. Kulatilleke, Mohanad Sarhan, and Marius Portmann. 2024. "FlowTransformer: A Transformer Framework for Flow-Based Network Intrusion Detection Systems." *Expert Systems with Applications* 241: 122564. doi:10.1016/j.eswa.2023.122564.
- Menezes, Nicolas Airoza Telles de, and Flávio Luis de Mello. 2021. "Flow Feature-Based Network Traffic Classification Using Machine Learning." *Journal of Information Security and Cryptography (Enigma)* 8(1): 12–16. doi:10.17648/jisc.v8i1.79.
- Nguyen, Loc Gia, and Kohei Watabe. 2022. "Flow-Based Network Intrusion Detection Based on BERT Masked Language Model." In *Proceedings of the 3rd International CoNEXT Student Workshop*, New York, NY, USA: ACM, 7–8. doi:10.1145/3565477.3569152.
- Russell, Stuart J, and Peter Norvig. 2016. *Artificial Intelligence: A Modern Approach*. Pearson.
- Sinadskiy, AN, and NA Domukhovskiy. 2020. "Statistical-Entropy Method of Searching for Fields in Traffic of Unknown Protocols." *Язык в сфере профессиональной коммуникации.—Екатеринбург*: 667–73.
- Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. "Attention Is All You Need." In *Advances in Neural Information Processing Systems*, , 5998--6008.
- Xian, Kehua. 2021. "An Optimized Recognition Algorithm for SSL VPN Protocol Encrypted Traffic." *Informatica* 45(6). doi:10.31449/inf.v45i6.3730.
- Zhang, Zhiwei, Shuhui Gong, Zhaoyu Liu, and Da Chen. 2023. "A Novel Hybrid Framework Based on Temporal Convolution Network and Transformer for Network Traffic Prediction." *PLOS ONE* 18(9): e0288935. doi:10.1371/journal.pone.0288935.

