



Snort Meets Transformers: Accelerating Transformer-Based Network Traffic Classification for Real-Time Performance

Downloaded from: <https://research.chalmers.se>, 2026-03-10 04:22 UTC

Citation for the original published paper (version of record):

Changrampadi, M., Almgren, M., Picazo-Sanchez, P. et al (2025). Snort Meets Transformers: Accelerating Transformer-Based Network Traffic Classification for Real-Time Performance. EUROSEC 2025 - Proceedings of the 2025 European Workshop on System Security. <http://dx.doi.org/10.1145/3722041.3723098>

N.B. When citing this work, cite the original published paper.



Snort Meets Transformers: Accelerating Transformer-Based Network Traffic Classification for Real-Time Performance

Mohamed Hashim Changrampadi
Chalmers University of Technology and University of
Gothenburg
Gothenburg, Sweden

Magnus Almgren
Chalmers University of Technology and University of
Gothenburg
Gothenburg, Sweden

Pablo Picazo-Sanchez
School of Information Technology, Halmstad University
Halmstad, Sweden

Ahmed Ali-Eldin
Chalmers University of Technology and University of
Gothenburg
Gothenburg, Sweden

Abstract

Transformer-based models have emerged as a powerful solution for network traffic classification, achieving high accuracy by autonomously learning patterns in raw traffic data. However, their high computational costs make real-time deployment impractical. In contrast, industry-proven tools like Snort and Suricata offer efficient network analysis but rely on manually crafted signatures, resulting in slower updates and limited adaptability to emerging threats.

In this work, we propose a cascading model that leverages the strengths of both approaches. During training, a transformer-based model learns traffic patterns, which are then extracted using SHAP analysis to enhance the knowledge base of a signature-based IDS. In deployment, the IDS handles routine classifications, while only complex cases are escalated to the transformer model. Our experiments combining the analysis of ET-BERT with Snort demonstrate a four-fold performance improvement over running only ET-BERT without compromising false positive or false negative rates.

CCS Concepts

• **Security and privacy** → *Network security*; • **Computing methodologies** → *Artificial intelligence*.

Keywords

Network Traffic Analysis, Network Pre-trained Models, Intrusion Detection Systems (IDS)

ACM Reference Format:

Mohamed Hashim Changrampadi, Magnus Almgren, Pablo Picazo-Sanchez, and Ahmed Ali-Eldin. 2025. Snort Meets Transformers: Accelerating Transformer-Based Network Traffic Classification for Real-Time Performance. In *The 18th European Workshop on Systems Security (EuroSec'25), March 30-April 3, 2025, Rotterdam, Netherlands*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3722041.3723098>

1 Introduction

Network traffic classification, particularly intrusion detection, is crucial for maintaining a secure and resilient network infrastructure. A fundamental aspect of this process is classifying benign and malicious packets, where malicious packets should be ultimately dropped or otherwise mitigated. Typically, such classification or filtering is performed using open-source Intrusion Detection Systems (IDS) tools like Snort [12] and Suricata [1].

The primary advantages of these systems are their lightweight nature and high throughput, measured for example in Packets Per Seconds (PPS), usually achieving a good detection accuracy while processing up to 250k PPS [18].

These approaches rely on predefined signatures, typically crafted by the community and fine-tuned by local experts to reduce false positives within a specific network infrastructure. However, detecting new attacks and identifying threats in encrypted traffic remains challenging, requiring more advanced methods such as machine learning models [8].

Supervised machine learning and deep learning-based IDS [8] promise to eliminate the need for human experts and enable the detection of sophisticated attacks with patterns too complex for manual signature creation. However, their effectiveness heavily relies on the availability of large and labeled datasets, often a scarce resource for cybersecurity.

Recent advancements in deep learning have enabled the use of pre-trained models, such as transformers, which leverage vast amounts of unlabeled data and require only a small labeled dataset for fine-tuning. This makes them a good fit for network security, where unlabeled traffic is abundant and can be effectively utilized for training. Such pre-trained models have been the state of the art technique in natural language processing [2], computer vision [3] and gradually introduced for web traffic classification [4, 5, 7, 16].

One such transformer-based model for network traffic classification is ET-BERT [7], which employs Bidirectional Encoder Representations from Transformers (BERT) pre-trained model to learn traffic patterns and performs fine-tuning on several downstream tasks including malware traffic classification. Several studies [4, 21] use ET-BERT as the state-of-the-art baseline and propose improvement with better accuracy and performance.

Despite showing robust classification performance on several datasets, these approaches has very low throughput. For instance, FastTraffic [19], reported the throughput of 1010 PPS, comparing



This work is licensed under a Creative Commons Attribution 4.0 International License. *EuroSec'25, March 30-April 3, 2025, Rotterdam, Netherlands*
© 2025 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1563-1/2025/03
<https://doi.org/10.1145/3722041.3723098>

to ET-BERT with 94 PPS on a high configuration device (10GB GPU). Hence, the performance of ET-BERT is over three orders of magnitude slower than a typical rule-based system [18].

In this work, we explore the possibility of integrating the high throughput analysis of a signature-based intrusion detection with the powerful and accurate analysis of a transformer-based model using pretraining and fine-tuning. More specifically, we propose a cascaded traffic classification framework that combines Snort with ET-BERT to achieve real-time performance without compromising detection accuracy. During the *training phase*, ET-BERT learns to classify network traffic, similar to previous approaches. However, we hypothesize that some cases are easier to classify than others. To leverage this, we use a model interpretability technique to extract part of the learned knowledge and validate it against the training set, generating distinct byte sequence patterns suitable for Snort signatures. These patterns are then added to the existing Snort signature base.

In the *testing phase*, Snort analyzes all traffic, handling the majority of classifications. Complex cases that do not match any signature are then forwarded to ET-BERT for final analysis. This allows a substantial portion of packets to be processed by Snort in a fraction of the time, reducing the load on the computationally intensive transformer model. As a result, the framework achieves significant speed improvements without sacrificing classification performance.

To summarize, our contributions are as follows:

- We propose a heuristic approach to extract highly distinguishable token patterns from transformer-based traffic classification models.
- We introduce a cascaded traffic classification framework that combines Snort for faster pattern matching with transformer-based models for accurate traffic classification, thereby substantially improving the throughput over 300%.
- We evaluate the proposed framework on a comprehensive dataset, demonstrating increased throughput and reduced computational overhead of transformer-based models.

2 Background

In this section, we provide background on ET-BERT, which is a transformer-based model designed for traffic classification, and explore model interpretability using SHAP (SHapley Additive exPlanations).

2.1 The ET-BERT

Transformer-based models can learn the complex patterns present in network traffic data. ET-BERT [7], specifically, is a pre-trained transformer model designed to convert raw network traffic into meaningful representations for accurate packet classification. It utilizes BERT to understand the contextual relationships within raw traffic bytes. The use of transformer-based models involves two primary stages: pre-training and fine-tuning. During pretraining, the model captures complex patterns and correlations in the data, while fine-tuning enables it to identify discriminative features for classifying data into specific categories. ET-BERT's methodology is similar to that of BERT [2]. However, a key difference is that, in

ET-BERT, byte-pair tokens are analogous to words, and packets (sequences of tokens) are analogous to sentences in Natural Language Processing (NLP).

The ET-BERT's traffic classification framework comprises three key steps: preprocessing, pretraining, and fine-tuning. During preprocessing, unlabeled packets from raw traffic data is processed and represented as a sequence of overlapping byte-pair tokens. By tokenizing the packet data, ET-BERT ensures that the model can learn from both low-level byte sequences and higher-level network patterns. During pretraining, ET-BERT learns the generic token representations from the large-scale unlabeled traffic data. The pretraining process leverages a self-supervised learning approach, where the model learns by predicting masked tokens within packet sequences. Once the model has learned a strong foundation of generic traffic representations, it undergoes fine-tuning on labeled traffic datasets. This step involves training ET-BERT to classify packets into specific categories, such as malware traffic classification, apps classification, etc. Fine-tuning refines the pre-learned representations, optimizing them for targeted classification tasks. By leveraging both the pretrained knowledge and the labeled dataset, ET-BERT achieves good accuracy and generalization.

2.2 Model analysis using SHAP

To leverage the complex learning capabilities of the ET-BERT model for identifying distinctive patterns, we use SHAP [11], a perturbation-based model interpretability approach. SHAP is used to quantify the contribution of each input feature—in our case, byte-pair tokens—to the model's predictions. This is achieved by perturbing the input and observing the effect on the model's inference. By observing the change in the model's prediction when a feature is absent, SHAP assigns a score to the feature, indicating its importance. A high SHAP value for a specific token suggests that its presence significantly increases the probability of the model inferring a particular class. SHAP-based model interpretability approaches have been applied to various machine learning models, including tree-based models [10] and deep learning models [14], and for different types of data such as images and text. In our work, we utilize SHAP to identify the most significant byte-pair tokens in network traffic sequences used by ET-BERT for traffic classification.

3 Methodology

The overall goal is to explore how the efficiency of rule-based systems can be combined with the accuracy of transformer-based models. We aim to utilize the ET-BERT's strong learning ability of distinctive tokens across classes, and extract these significant tokens using SHAP. By utilizing these significant tokens, we generate a distinctive byte sequence pattern for faster inference through Snort, reducing the load on the low throughput ET-BERT.

3.1 Proposed Architecture

Figure 1 depicts our proposed architecture. In the training phase (left), we use transformer-based methods to extract patterns that are specific to each class to form the basis of an additional Snort rule knowledge base. In the testing phase (right), these new rules allow Snort to classify a majority of (the simpler) cases, leaving a minority for the more powerful but slower ET-BERT.

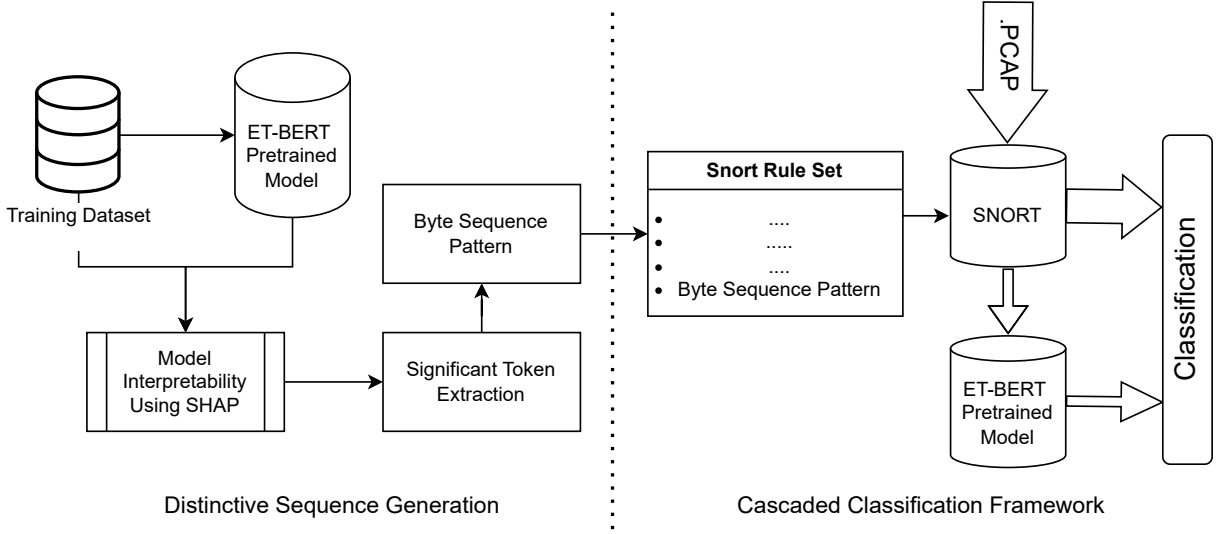


Figure 1: Proposed architecture, showing the two-tier classification framework.

Below we outline the methodology to extract the significant tokens and the creation of sequences that form the basis for rules that work well with Snort.

Algorithm 1 SHAP-Based Significant Token Extraction

Require: Pre-trained Transformer model M , Training set $D = \{p_1, p_2, \dots, p_N\}$, where p_i is a sequence of byte-pair tokens with class label $y_i \in \{1, 2, \dots, C\}$, C is the number of classes.

Ensure: Significant tokens T_c for each class c .

- 1: **Step 1:** Construct a baseline packet p_{base} with a random sequence of byte-pair tokens.
 - 2: **Step 2:** Compute SHAP values for all samples in D using KernelSHAP.
 - 3: **for** each packet $p_i \in D$ **do**
 - 4: Generate perturbed packets $\mathcal{P}_i^* = \{p_{i1}^*, p_{i2}^*, \dots\}$ by replacing subsets of tokens in p_i with tokens from p_{base} .
 - 5: **for** each perturbed packet $p_{ij}^* \in \mathcal{P}_i^*$ **do**
 - 6: Compute model output $y_{ij}^* = M(p_{ij}^*)$ for all classes.
 - 7: **end for**
 - 8: Compute SHAP values ϕ_i for each token in p_i based on model outputs.
 - 9: **end for**
 - 10: **Step 3:** Aggregate SHAP values for all unique tokens in D .
 - 11: **for** each unique token t in D **do**
 - 12: Compute aggregate SHAP value $\Phi_t = \sum_{p \in D} \phi_t(p)$, where $\phi_t(p)$ is the SHAP value of token t in packet p .
 - 13: **end for**
 - 14: **Step 4:** Identify significant tokens for each class.
 - 15: **for** each class $c \in \{1, 2, \dots, C\}$ **do**
 - 16: Select top S tokens $T_c = \{t_1, t_2, \dots, t_S\}$ with the largest Φ_t for class c .
 - 17: **end for**
 - 18: **Return:** Significant tokens T_c for each class c .
-

3.2 Significant Token Extraction

We use SHAP to identify the most significant byte-pair tokens in network traffic sequences for classification. To efficiently compute SHAP values, we employ KernelSHAP, a model-agnostic approximation method based on Shapley values and Linear LIME. For each packet in the training set, KernelSHAP generates perturbed inputs by replacing subsets of tokens with tokens from a baseline packet. This baseline packet consists of a random sequence of byte-pair tokens and is used as a neutral reference. The model's output is then computed for each perturbed input, and SHAP values are derived to quantify the contribution of each token position to the classification decision. The aggregate SHAP values are then computed for each unique token across all packets in the training set. The details of extraction of significant tokens is outlined in Algorithm 1.

Tokens with the largest aggregate SHAP values are identified as the most significant for each class. Specifically, we select the top 10 significant tokens per class based on their SHAP values. In Figure 2, we show a list of the top 5 significant tokens for class 0, along with their count across all other classes. We can observe that the `0a11` tokens predominantly exists only in the target class 0; however, others (`0300`, `0005`, `080a`, and `0000`) predominantly exists in other classes. While the token `0a11` predominantly appears in class 0, it also occurs in other classes, though infrequent, making it a suboptimal choice as a distinct token for class 0. Therefore, we analyze the positions of these significant tokens (see bottom in Figure 2) to heuristically identify a distinct token sequence pattern that could reliably represents the specific class.

3.3 Distinctive Sequence Generation

The most significant tokens along with their positions, provide crucial information for generating class-discriminative sequences. Each significant token is paired with its most common position in the training set. This mapping captures not only the importance of the token but also its typical location within the input sequence.

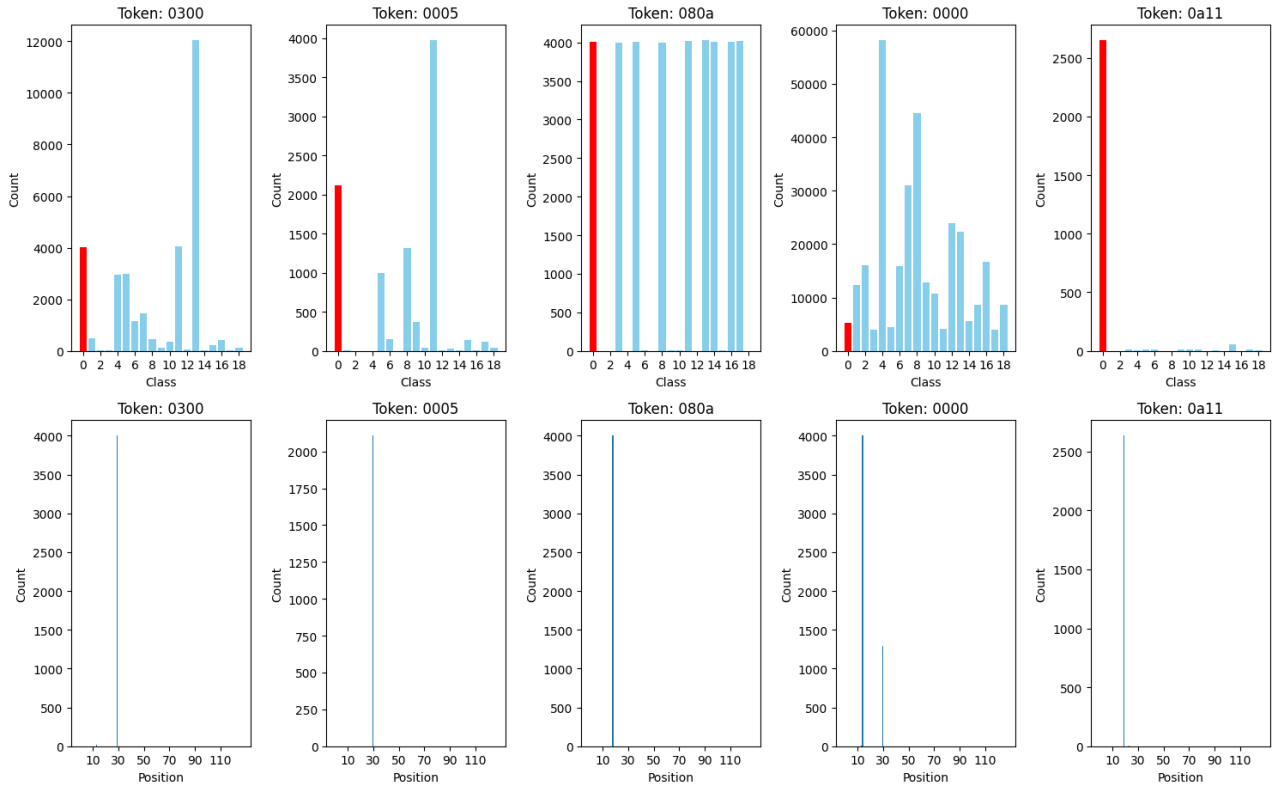


Figure 2: Top 5 significant tokens of a benign class 0 (BitTorrent), their count across all other classes (top); their aggregate positions in the training set (bottom)

By associating tokens with their positions, we gain insights into the structural patterns that the model relies on for classification.

Algorithm 2 outlines the generation of distinctive byte sequences. The algorithm first identifies the most common position of each significant token within its class and pairs them accordingly. It then splits byte-pair tokens into individual bytes, assigning positions based on the original token. These bytes are grouped into continuous sequences and validated against the training data. Only sequences exclusive to the target class are selected as distinctive patterns. Finally, these unique byte sequences are used to generate Snort rules for pattern matching.

3.4 Cascaded Classification Framework

The cascaded classification framework would operate in two layers. The goal is to reduce the computational load on the transformer-based model, by using the much faster Snort to process the majority of the traffic, and only leveraging ET-BERT for the smaller subset of remaining packets.

In the first layer, incoming network packets are processed by Snort, which uses the generated ruleset to classify packets based on the extracted byte sequence patterns. Packets matching these distinctive sequences are filtered or classified, while those that do not match any rules are forwarded to the second layer. This initial filtering process by Snort should significantly reduce the

workload on the computationally intensive deep learning model if the patterns are comprehensive enough.

The second layer utilizes the ET-BERT model to analyze and classify the remaining unmatched packets with high accuracy.

4 The Experimentation and Results

In this section, we present the details for training and evaluating ET-BERT, followed by an extensive evaluation done of the cascaded classification framework.

4.1 Datasets

To train and validate our models, we use two datasets, CIC-IDS-2017 and USTC-TFC.

CIC-IDS-2017 [13]: The dataset consists of raw pcap files capturing five days (Mon-Fri) of network traffic during working hours. We use the full dataset (50GB) for pretraining, unlike [7] that use only 15GB (see Section 4.2). We only use this dataset for pre-training.

USTC-TFC [17]: This dataset originally comprises ten benign classes and ten malicious traffic classes, totaling 3.8GB. However, for our experiments we remove the malicious class “Tinba” as there remains very few packets after the preprocessing step.

Algorithm 2 Distinctive Byte Sequence Extraction

Require: Training set $D = \{p_1, p_2, \dots, p_N\}$, where p_i is a sequence of byte-pair tokens with class label $y_i \in \{1, 2, \dots, C\}$, C is the number of classes; Significant tokens T_c for each class c .

Ensure: Distinctive byte sequence patterns U_c for each class c .

- 1: **Step 1:** $\forall t \in T_c$, compute its most common position $p_t^* = \arg \max_{p \in P_t} \text{frequency}(p)$, where P_t is the set of positions where t appears in class c .
- 2: **Step 2:** Pair each token $t \in T_c$ with its common position p_t^* .
- 3: **Step 3:** Split each byte-pair token t into two bytes b_1 and b_2 , with positions $p_1 = p_t^*$ and $p_2 = p_t^* + 1$, respectively.
- 4: **Step 4:** Extract byte sequences with continuous positions for each class c .
- 5: **for** each class c **do**
- 6: Collect all bytes $b_{1,2,\dots,j}$ and their positions $p_{1,2,\dots,j}$ from the paired tokens in T_c .
- 7: Group bytes into sequences where positions are continuous
- 8: **end for**
- 9: **Step 5:** Generate unique byte sequences $U_c = \{u_1, u_2, \dots\}$ for each class c , where each u_i is a continuous byte sequence extracted in Step 4.
- 10: **Step 6:** Validate and select distinctive byte sequence patterns.
- 11: **for** each class c **do**
- 12: Validate each sequence $u_i \in U_c$ using the training set D .
- 13: Select sequences that appear only in the target class c and discard those appearing in other classes.
- 14: Update U_c to include only the distinctive sequences.
- 15: **end for**
- 16: **Return:** Distinctive byte sequence patterns U_c for each class c .

For the validation of our ET-BERT implementation described in Section 4.2, we use the same methodology as in the original paper and we create a balanced small dataset consisting of 5000 packets per class, allocating 80% for training, 10% for validation, and 10% for testing. For our other experiments, we use the remaining 2.7 million packets. This allows us to use as much as possible of the labeled dataset for testing, but with an unbalanced class distribution. For a security-focused analysis, we further organize the data into two clusters: one comprising the ten benign traffic classes and the other comprising the nine malicious traffic classes. As will be seen from the experiments, the accuracy is very high so we report false positives and false negatives to distinguish between the cases.

4.2 Reproducing ET-BERT: Model Validation

ET-BERT is implemented following a similar approach to that described in the original paper. The pretraining dataset used in ET-BERT is partially public, therefore we use only the publicly available dataset for pretraining, i.e., we use CIC-IDS-2017 for pretraining and USTC-TFC for fine-tuning on malware traffic classification.

We follow the original implementation [7] by explicitly defining preprocessing, pretraining, and fine-tuning steps for consistency. We exclude packets using ARP, DNS, DHCP, ICMP, and IGMP protocols. As in the original approach, we remove the first 36 bytes of each packet, including the Ethernet header, IP addresses, and

ports. To ensure sufficient data for token generation, we discard packets smaller than 40 bytes.

After pretraining, we fine-tune the model using USTC-TFC for the malware classification task. We use 5,000 packets per class from the USTC-TFC dataset. The training, validation, and test sets follow an 80:10:10 split. Table 1 presents validation of the original model and our implementation of ET-BERT on the USTC-TFC test set. The slight performance drop in our implementation is likely due to the variation in the pre-training dataset.

4.3 The Cascaded Classification Framework

To evaluate the cascaded classification framework, we need distinctive sequences to use as a ruleset in Snort. To extract significant tokens and generate distinctive byte sequence patterns, we use the training set. The training set consists of two major classes: benign and malicious. The benign class includes 10 subclasses, while the malicious class has 9. We extract the top 10 significant tokens for each subclass to generate distinctive byte sequence patterns. In total, we generate 27 sequences for the benign class and 9 sequences for the malicious one. These patterns are fed into Snort as the first layer of the classification framework.¹

We evaluate the framework on the comprehensive test set of 2.7 million packets, initially processed by Snort. After pattern matching, Snort provides both the matched and the unmatched packets. We then ran the unmatched packets using ET-BERT for further classification. Figure 3 presents the percentage and count of packets matched and unmatched by Snort for benign, malicious, and overall traffic. The performance of the overall framework, along with layer-wise performance, is compared to the standalone ET-BERT in Table 2. We evaluate benign and malware classification by analyzing the absolute number of False Positives (FPs), False Negatives (FNs), and throughput (TH) in PPSs.

The performance of standalone ET-BERT on the comprehensive test set, as listed in Table 2, results in 6 FPs and 4 FNs while processing 100% of packets with a throughput of 600 PPSs. In the proposed approach, the entire comprehensive test set is first processed by Snort. The rule-based classification in Snort identifies 71.9% of the packets matching the distinctive pattern, requiring only 28.1% of unmatched packets to be analyzed by ET-BERT for further classification. For these 71.9% of filtered packets, Snort achieves 100% accuracy with zero false positives and false negatives.

ET-BERT then processes the remaining 28.1% of the packets, achieving an accuracy of >0.999 with four FPs and three FNs. Since all traffic passes through Snort before unmatched packets are forwarded to ET-BERT, there is an overhead of processing some packets twice. However, as Snort processes packets significantly faster than ET-BERT, this overhead is negligible.

Overall, the cascaded classification approach increases throughput by more than four times compared to the standalone approach, without increasing FPs nor FNs. From a security perspective, combining rule-based Snort with ET-BERT maintains FP/FN rates while improving processing efficiency, demonstrating the robustness of the distinctive patterns generated from the model.

¹Since Snort does not provide a built-in solution for detecting byte patterns in packet headers, we developed custom plugins to detect specific byte patterns spanning across packet headers.

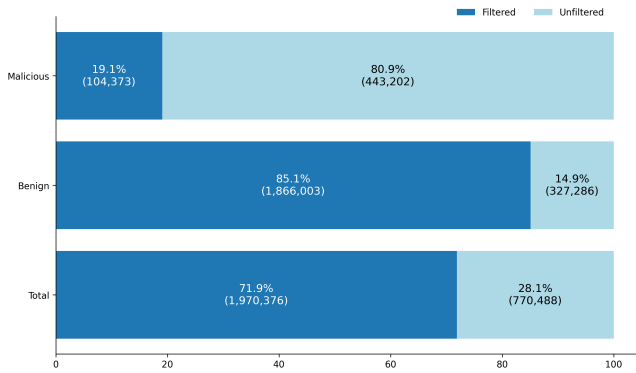
Model	Accuracy	USTC-TFC		
		Precision	Recall	F ₁
ET-BERT (reported)	0.9915	0.9915	0.9916	0.9916
ET-BERT	0.9828	0.9837	0.9828	0.9829

Table 1: Performance of ET-BERT [7] vs our version

Configuration	Packets (%)	Packet (millions)	FP	FN	TH (PPS)
Standalone ET-BERT	100%	2.74	6	4	0.6k
Cascaded (Overall)	128.1%	3.51	4	3	2.7k
Layer 1 (Snort)	100%	2.74	0	0	152.2k
Layer 2 (ET-BERT)	28.1%	0.77	4	3	0.6k

Table 2: Performance Comparison of Standalone and Hybrid Approaches

Note: In the cascaded framework, all packets (100%) are first processed by Snort. Then, only 28.1% of those packets are forwarded to ET-BERT for further analysis. Thus, if you sum both stages, the cumulative processing percentage is 128.1%.

**Figure 3: Snort filtered and unfiltered packets for benign and malicious class**

5 Discussion

Our pattern extraction is conservative, to not increase FPs or FNs. From Table 2, we can see we do not have any misclassification (FP/FN) in the Snort stage. We examined subclass-level misclassifications within benign and malicious categories to further analyze classification performance. Our analysis revealed 47 misclassifications, all occurring between different malicious subclasses. This indicates that incorporating Snort-based classification does not negatively impact the overall model accuracy, although subclasses might have misclassifications. In this work, we only use byte sequence patterns in the Snort ruleset for filtering. However, integrating Snort’s existing rich default ruleset alongside these byte sequence patterns would likely enhance performance further.

The comprehensive test set comprises 2.2 million benign packets and 0.5 million malicious packets. As shown in Figure 3, Snort triggers on 85% of benign traffic and 19% of malicious traffic. In real-world scenarios, benign traffic significantly dominates malicious traffic. Under these conditions, Snort would theoretically filter 85%

of the packets, forwarding only 15% to ET-BERT, thereby significantly reducing the transformer model’s computational workload by a factor of six. Meanwhile, since Snort primarily filters benign traffic, most of malicious packets are forwarded to ET-BERT for precise classification, ensuring robust security.

6 Related Work

Transformer-based models have gained significant attention in network traffic classifications [4, 5, 7, 21] due to their ability to capture complex patterns. Flow-MAE [4] proposed further improvement to ET-BERT [7], reducing memory overhead and computational complexity. Other approaches, such as CNN-based malware classification [17] have also been explored. Unlike transformers, CNN-based methods treat traffic data as images, removing the need for expert features for training. Because of the complex learning capability of these approaches, the computational complexity of the model results in low throughput, despite being robust.

Few studies have also explored hybrid approaches to network classification, combining different techniques to improve performance and accuracy. For instance, a hybrid intrusion detection system [20] combines the strength of misuse and anomaly detection approaches. It uses the random forests algorithm for both components; misuse detection identifies known attacks, while anomaly detection, through outlier detection, identifies novel intrusions. Another hybrid approach [6] combines signature-based detection (Snort) with anomaly-based detection (Naive Bayes) to detect attacks, overcoming limitations of individual approaches. A two-stage system [15] employs Snort for initial malicious activity detection, followed by flow-based anomaly detection system with Deep Neural Networks (DNNs). However, in our work, while integrating Snort, we utilize the model learning capability to extract the significant token and use in the Snort ruleset, providing rich granular control over the detection, along with default Snort capabilities.

Model interpretability is crucial in Deep Learning (DL)-based network security applications. To improve trustworthiness of the AI-based security systems, Zhi et al. [9] propose novel model-agnostic explanation method that identifies the relevance of specific features for malware detection. In our work, we addressed the low throughput of current robust transformer models with extraction of significant tokens using SHAP, integrating with industry-proven high throughput tools like Snort with explainable ruleset.

7 Conclusion

In this paper, we demonstrated the effectiveness of using transformer-based models in extracting significant tokens for classification. The proposed cascade approach improves the throughput compared to standalone methods without a drop in performance.

Combining the analysis of rule-based Snort with ET-BERT does not increase false positives nor false negatives, implying the robustness of the extracted distinctive patterns. Our approach achieves a threefold increase in throughput over the standalone method without compromising accuracy. This combination, with a focus on strict security measures, reinforces the reliability of the extracted patterns, ensuring effective and fast traffic classification.

Acknowledgments

The computations was enabled by resources provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

References

- [1] Albin, E., Rowe, N.C.: A Realistic Experimental Comparison of the Suricata and Snort Intrusion-Detection Systems. In: 2012 26th International Conference on Advanced Information Networking and Applications Workshops. p. 122–127. IEEE (2012)
- [2] Devlin, J.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018). <https://doi.org/10.48550/arXiv.1810.04805>
- [3] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In: ICLR 2021: 9th International Conference on Learning Representations (2021)
- [4] Hang, Z., Lu, Y., Wang, Y., Xie, Y.: Flow-MAE: Leveraging Masked AutoEncoder for Accurate, Efficient and Robust Malicious Traffic Classification. In: Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses. p. 297–314. RAID '23, Association for Computing Machinery, New York, NY, USA (2023). <https://doi.org/10.1145/3607199.3607206>
- [5] He, H.Y., Guo Yang, Z., Chen, X.N.: Pert: Payload encoding representation from transformer for encrypted traffic classification. In: 2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K). pp. 1–8 (2020). <https://doi.org/10.23919/ITUK50268.2020.9303204>
- [6] Hussein, S.M., Mohd Ali, F.H., Kasiran, Z.: Evaluation Effectiveness of Hybrid IDS Using Snort with Naive Bayes to Detect Attacks. In: 2012 Second International Conference on Digital Information and Communication Technology and Its Applications (DICTAP). p. 256–260. IEEE (2012)
- [7] 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. In: Proceedings of the ACM Web Conference 2022. pp. 633–642 (2022)
- [8] Lotfollahi, M., Jafari Siavoshani, M., Shirali Hossein Zade, R., Saberian, M.: Deep Packet: A Novel Approach for Encrypted Traffic Classification Using Deep Learning. *Soft Computing* 24(3), 1999–2012 (2020)
- [9] Lu, Z., Thing, V.L.: How Does It Detect a Malicious App? Explaining the Predictions of AI-based Malware Detector. In: 2022 IEEE 8th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS). p. 194–199. IEEE (2022)
- [10] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I.: From Local Explanations to Global Understanding with Explainable AI for Trees. *Nature Machine Intelligence* 2(1), 56–67 (2020)
- [11] Lundberg, S.M., Lee, S.I.: A Unified Approach to Interpreting Model Predictions. In: Guyon, I., Luxburg, U.V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., Garnett, R. (eds.) *Advances in Neural Information Processing Systems* 30, p. 4765–4774. Curran Associates, Inc. (2017)
- [12] Roesch, M.: Snort: Lightweight intrusion detection for networks. In: LISA. vol. 99, p. 229–238 (1999)
- [13] Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* 1, 108–116 (2018)
- [14] Shrikumar, A., Greenside, P., Kundaje, A.: Learning Important Features Through Propagating Activation Differences. In: *International Conference on Machine Learning*. p. 3145–3153. PMLR (2017)
- [15] Ujjan, R.M.A., Pervez, Z., Dahal, K.: Suspicious Traffic Detection in SDN with Collaborative Techniques of Snort and Deep Neural Networks. In: 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). p. 915–920. IEEE (2018)
- [16] Wang, T., Xie, X., Wang, W., Wang, C., Zhao, Y., Cui, Y.: NetMamba: Efficient Network Traffic Classification via Pre-training Unidirectional Mamba. In: The 32nd IEEE International Conference on Network Protocols (ICNP 2024), Charleroi, Belgium (October 2024), october 28–31, 2024
- [17] Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware Traffic Classification Using Convolutional Neural Network for Representation Learning. In: 2017 International Conference on Information Networking (ICOIN). p. 712–717. Da Nang, Vietnam (2017). <https://doi.org/10.1109/ICOIN.2017.7899588>
- [18] White, J.S., Fitzsimmons, T., Matthews, J.N.: Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata. In: *Cyber Sensing 2013*. vol. 8757, p. 10–21. SPIE (2013)
- [19] Xu, Y., Cao, J., Song, K., Xiang, Q., Cheng, G.: FastTraffic: A Lightweight Method for Encrypted Traffic Fast Classification. *Computer Networks* 235, 109965 (2023)
- [20] Zhang, J., Zulkernine, M.: A Hybrid Network Intrusion Detection Technique Using Random Forests. In: *First International Conference on Availability, Reliability and Security (ARES'06)*. pp. 8–pp. IEEE (2006)
- [21] Zhao, R., Zhan, M., Deng, X., Wang, Y., Wang, Y., Gui, G., Xue, Z.: Yet Another Traffic Classifier: A Masked Autoencoder Based Traffic Transformer with Multi-Level Flow Representation. *Proceedings of the AAAI Conference on Artificial Intelligence* 37(4), 5420–5427 (2023)