Intelligent Classification Techniques for Detecting Malicious Network Traffic

1st Ilham Aliyev
Department of Computer Engineering
Istanbul Arel University
Istanbul, Turkey

2nd Mert Bahadır
Department of Computer Engineering
Istanbul Arel University
Istanbul, Turkey

***Abstract*—Detecting malicious network traffic has become increasingly challenging due to the widespread adoption of encryption, which obscures packet content from conventional network traffic analysis tools. In this paper, we replicate and evaluate the effectiveness of the Time Series Feature Network (TSFN [2]), a classification framework originally proposed for detecting malicious network activity in encrypted traffic environments. TSFN combines a BERT-based contextual embedding model with a Long Short-Term Memory (LSTM) network for time series analysis, capturing both packet-level and flow-level representations. This hybrid architecture enables the model to learn global patterns and temporal dependencies, enhancing detection accuracy and robustness against evolving cyber threats. Through extensive experimentation using the USTC-TFC dataset, we validate that TSFN [2] outperforms baseline models, including both traditional deep learning approaches and recent pre-trained models. The replication results confirm the advantages of TSFN for accurate malicious traffic detection, making it a valuable tool for network security in modern encrypted environments.**

***Index Terms*—Malicious traffic detection, BERT [5], LSTM, encrypted connections, machine learning, network security, anomaly detection.**

I. Introduction

The exponential rise in the volume of encrypted network traffic has intensified the challenge of detecting malicious activity within network communications. Traditional methods for detecting network anomalies and attacks, such as Deep Packet Inspection (DPI) and rule-based signature detection, are increasingly ineffective against encrypted traffic. This is because such methods rely on inspecting the payload of packets, which is hidden within encrypted protocols. As a result, network defenders are left with limited tools for inspecting encrypted traffic, and attackers are able to exploit encryption as a means of evading detection.

To address this problem, researchers have turned to machine learning (ML) and deep learning (DL) models, which can analyze traffic metadata such as packet size, timing, and flow patterns, without needing access to packet content. These models can detect anomalous traffic patterns based on flow-level features and temporal dynamics, making them an essential tool for identifying malicious traffic within encrypted communications.

In this paper, we introduce the Time Series Feature Network (TSFN) [2] alongside ET-BERT [8], an advanced pre-trained model tailored for encrypted traffic classification. ET-BERT [8] leverages datagram-level pre-training on large-scale datasets, enabling it to learn robust representations even without direct access to packet content. The ET-BERT [8] model includes two innovative pre-training tasks, the Masked BURST Model (MBM) and Same-origin BURST Prediction (SBP), designed specifically for encrypted traffic. MBM masks tokens within datagram sequences and tasks the model with predicting these masked positions based on context, capturing byte-level dependencies critical to understanding encrypted data patterns. SBP, on the other hand, captures transmission order and packet-level relationships by distinguishing between BURSTs within a flow. Together, these tasks enable ET-BERT [8] to generalize across a range of encryption protocols and traffic types.

Additionally model, TSFN [2], combines a BERT-based contextual encoder with Long Short-Term Memory (LSTM) layers to capture both static (global) and dynamic (temporal) features of network traffic. This hybrid approach allows TSFN to effectively analyze both packet-level patterns and flow-level dependencies, which is crucial for accurate detection of encrypted malicious traffic. The experimental evaluation using the USTC-TFC dataset demonstrates that TSFN [2] outperforms several baseline models, including classical deep learning models as well as recent pre-trained architectures, in terms of F1 score and detection accuracy. We show that TSFN [2] represents a significant advancement in the field of network traffic analysis and detection, particularly in environments where traditional techniques struggle to perform.

The remainder of this paper is structured as follows: Section II reviews existing work in the area of network traffic classification and anomaly detection. Section III outlines the methodology behind TSFN [2], and ET-BERT[8] including its architecture and key components. Section IV presents the experimental setup, results, and comparisons with baseline models. Finally, Section V discusses the implications of our findings, identifies the limitations of the current approach, and suggests directions for future work.

II. Related Work

A. Traditional Network Security Approaches

Historically, network security approaches have relied heavily on methods such as Deep Packet Inspection (DPI) and signature-based detection. DPI involves scanning the content of each packet to detect known attack signatures, which is effective

for identifying malicious traffic patterns. However, DPI is not feasible when traffic is encrypted, as the packet payload is unreadable. Signature-based detection also faces similar challenges, as it can only detect known attack patterns and is unable to identify new or evolving threats, especially in the context of encrypted traffic. As encryption becomes more widely adopted, these traditional methods are increasingly being rendered obsolete in modern cybersecurity applications.

B. Feature Engineering in Network Security

Feature engineering plays a pivotal role in machine learning-based traffic classification. Typical features used in network traffic analysis include packet size, flow duration, byte counts, and the time intervals between packets. While manual feature engineering can be effective in capturing certain traffic patterns, it is often limited in its ability to adapt to new types of attacks or encryption schemes. This is particularly true when dealing with encrypted traffic, where traditional feature sets may not provide sufficient information for accurate classification. In this context, deep learning methods, which can automatically extract features from raw traffic data, have gained prominence as a more flexible and adaptive alternative.

C. Machine Learning and Deep Learning in Traffic Classification

Machine learning techniques, including Decision Trees, Random Forests, and Support Vector Machines (SVM), have been widely used in traffic classification. These models focus on metadata such as packet sizes and flow durations, using this information to classify traffic without needing access to the packet content. Recent advancements in deep learning, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have further improved the performance of traffic classification models. CNNs are particularly effective at learning patterns in raw traffic data, while RNNs excel at modeling sequential dependencies in network flows.

Despite these improvements, however, the rise of encrypted traffic presents new challenges, as traditional models struggle to capture the temporal dynamics and complex patterns in encrypted flows. The integration of contextual feature extraction methods, such as those provided by BERT [5] models, represents a promising avenue for improving the classification of encrypted traffic.

D. Challenges of Encrypted Traffic Classification

Encrypted traffic represents a significant challenge for traditional traffic classification methods, which are often unable to detect malicious activity without access to packet payloads. Recent approaches like FlowPrint and Deep Fingerprinting aim to classify encrypted traffic based on flow-level metadata. These methods have shown promise, but they are still limited by their reliance on static feature representations that do not adapt well to changing traffic patterns. This limitation becomes particularly evident when dealing with dynamic threats that evolve over time.

E. Pre-trained Models in Encrypted Traffic Detection

Transformer-based models such as BERT [5] have demonstrated significant success in natural language processing tasks by leveraging self-attention mechanisms to capture rich contextual information. Recent work has applied BERT to network traffic analysis, using it to extract contextualized representations of traffic data for classification tasks. Models such as ET-BERT [8] and PERT utilize BERT for encrypted traffic classification, pre-training on large-scale datasets to learn generalizable traffic representations. However, these models primarily focus on packet-level features and tend to overlook the temporal dependencies that are critical for modeling sequential traffic patterns. In contrast, TSFN model integrates BERT with LSTM layers to address both global and sequential features in traffic data, improving performance in encrypted environments.

III. Existing Methodology: TSFN [2] - Time Series Feature Network

A. Model Architecture

The Time Series Feature Network (TSFN [2]) is a novel approach designed to classify encrypted network traffic by effectively capturing both static global patterns and dynamic temporal patterns within the data. The architecture of TSFN [2] combines two powerful components: a BERT-based packet encoder and an LSTM-based time series encoder. Together, these components enable the model to process network traffic at different levels of abstraction, from individual packet-level features to broader temporal patterns that unfold over time.

- **Packet Encoder (BERT-based):** The packet encoder utilizes a BERT-based architecture to transform each individual network packet into a high-dimensional embedding vector. BERT [5] , originally designed for natural language processing tasks, is adapted here to understand the contextual relationships between packets. This allows the model to capture global features, such as traffic flow characteristics, packet size distributions, and correlations between packets that could indicate suspicious or anomalous behavior. The use of BERT [5] enables the model to effectively handle complex relationships between packets and recognize higher-level patterns that may not be immediately apparent through traditional analysis techniques.
- **Time Series Encoder (LSTM):** After encoding the packets, the sequence of embeddings is passed to an LSTM (Long Short-Term Memory) network. LSTM is a type of recurrent neural network that excels at capturing sequential dependencies and learning from temporal patterns in data. By processing the sequence of packet embeddings generated by the BERT [5] encoder, the LSTM layer can detect subtle sequential relationships that could indicate malicious activities. The combination of LSTM with BERT allows TSFN [2] to track long-term dependencies in traffic, which is crucial for detecting more sophisticated and time-dependent network attacks.

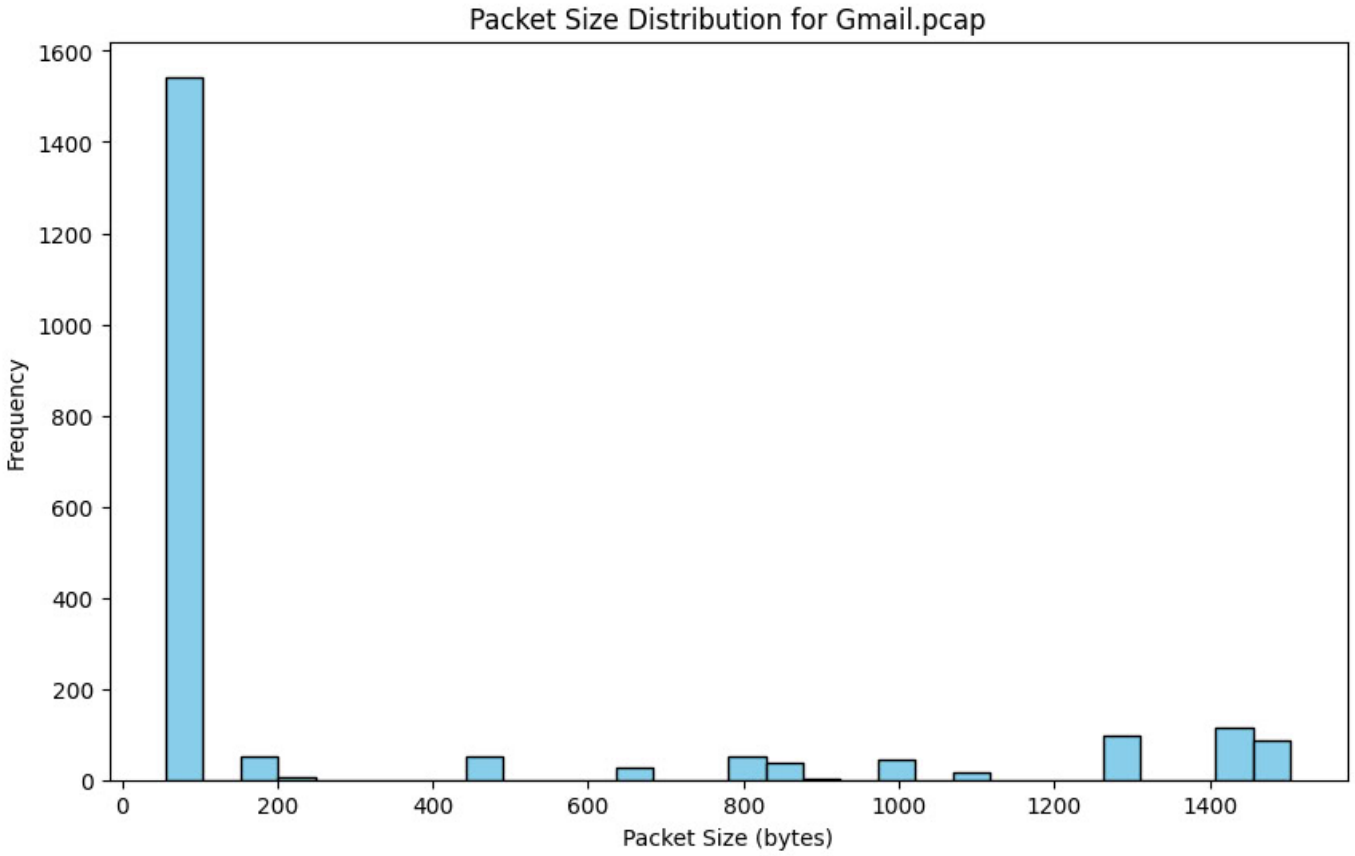

Fig. 1: Packet size distribution in one of the data classes

- **Masked BURST Model (MBM):** The Masked BURST Model (MBM) is a pre-training task within ET-BERT, inspired by the Masked Language Model (MLM) in traditional BERT frameworks, but adapted specifically for encrypted traffic classification. In natural language processing, MLM trains a model to predict randomly masked words based on their surrounding context, helping the model understand contextual relationships between words. Similarly, in MBM, selected tokens within datagram sequences are randomly masked, and the model is tasked with predicting these masked tokens based on the remaining context within the datagram. By learning dependencies among datagram bytes through this masking approach, MBM enables ET-BERT to capture complex, implicit relationships among packets, even in encrypted formats where content is hidden. This adaptation is critical for encrypted traffic, as it allows ET-BERT to build robust representations of patterns that are otherwise concealed, such as byte-level relationships unique to specific encryption protocols and flow patterns within secure network communications. Consequently, MBM enhances ET-BERT's ability to recognize intricate, protocol-specific byte dependencies across diverse traffic scenarios, which is essential for high accuracy in encrypted traffic classification [8].
- **Same-origin BURST Prediction (SBP):** The Same-origin BURST Prediction (SBP) task is another specialized pre-training task within ET-BERT, designed to capture the temporal and structural relationships inherent to network traffic. Encrypted network traffic typically consists of sequences of packets, often grouped into "BURSTs" that represent clusters of packets from a single source or destination within a given flow. SBP leverages this structure by requiring the model to learn whether two sub-BURSTs (sub-sections of BURSTs) originate from the same BURST within a flow. To accomplish this, the model is trained with pairs of sub-BURSTs, where some pairs are from the same BURST and others are not. The model must predict whether these pairs share a common origin, forcing it to learn the nuanced packet-level dependencies and transmission orders unique to each type of traffic flow. This focus on BURST structure is particularly valuable for encrypted traffic, where higher-level sequence patterns (such as the order and frequency of packet exchanges) play a vital role in distinguishing between different types of encrypted communications. SBP's ability to model such structural patterns makes ET-BERT resilient across a range of encryption protocols and enables it to generalize more effectively, thereby improving classification accuracy even with limited labeled data [8].

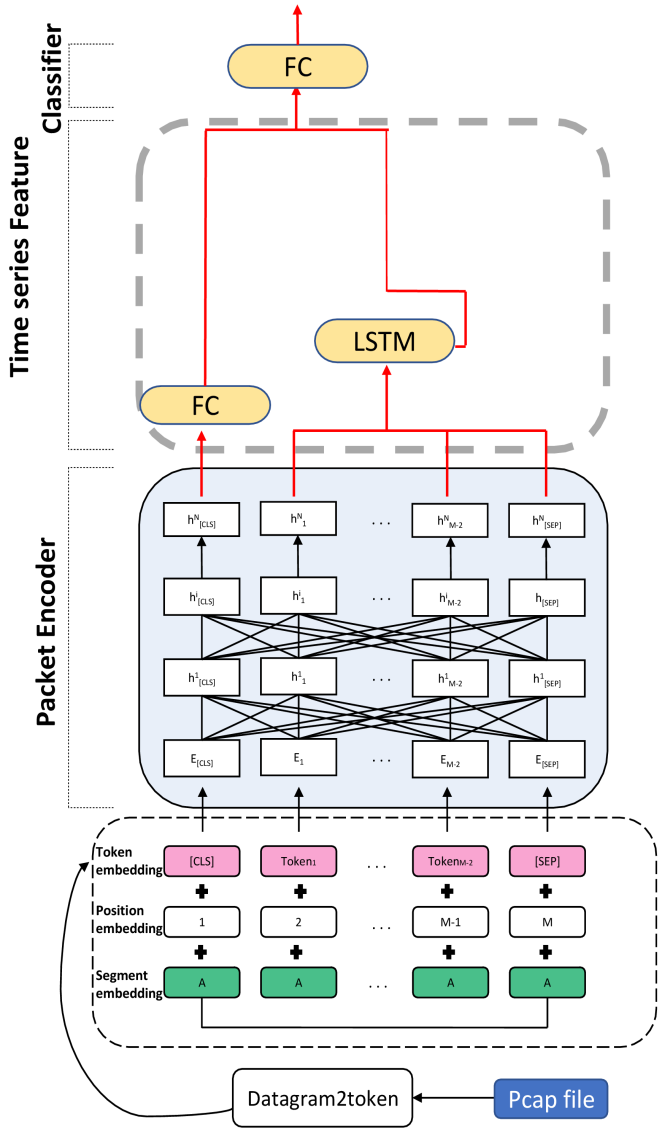

Fig. 2: Architecture of the Time Series Feature Network (TSFN [2])

The dual-layered approach of combining the BERT encoder with the LSTM encoder makes TSFN [2] highly effective in understanding encrypted network traffic. The packet encoder focuses on extracting high-level traffic features, while the time series encoder identifies temporal patterns, enabling TSFN [2] to detect both static and dynamic characteristics of the traffic. This hybrid model approach makes TSFN [2] a powerful tool for encrypted traffic classification, even in the presence of sophisticated, evolving attack strategies.

B. Dataset and Preprocessing

The USTC-TFC dataset serves as the foundation for training and evaluating TSFN [2]. This dataset is particularly suitable for multi-class classification tasks as it contains both benign and malicious traffic from a variety of encrypted application protocols, such as HTTPS, FTP, and VPN. The dataset includes several well-known malware families, such as Zeus, Mirai, and Tinba, as well as normal traffic from applications like web browsing and file transfers. The rich diversity of traffic types in the dataset allows the model to learn to distinguish between different traffic patterns, making it ideal for the task of encrypted traffic classification.

Before feeding the data into the model, it apply a series of preprocessing steps to ensure that the raw packet data is in a suitable format for training. The first step involves converting the raw packet data into a hexadecimal string representation, which simplifies the raw byte sequences into a human-readable format that preserves the relevant information. Additionally, it anonymize sensitive data by applying techniques like Datagram2Token, which replaces sensitive fields (e.g., IP addresses and ports) with random tokens. This ensures that the model can learn meaningful patterns without exposing private information, making the preprocessing pipeline crucial for maintaining privacy while still allowing effective model training.

Optimizing model performance also requires an understanding of the underlying data features. Packet size is a crucial aspect of network traffic and can range greatly amongst various data classes. We examine its distribution in one of the data classes to gain a better understanding of the connection between packet size and model performance. This packet size distribution is shown in Fig. 1, which provides information about the patterns found in the data. We may better customize the model's feature engineering and preprocessing pipeline to increase classification accuracy by looking at how packet sizes differ between malicious and benign traffic.

TABLE I: Dataset details.

Traffic Type	Application
Malicious traffic	Htbot, CridexNeris, Nsis-ay, Shifu, Virut, Zeus, Tinba, Miuref, Geodo
Normal traffic	Outlook, BitTorrent, FTP, Warcraft, MySQL, Skype, Facetime, SMB, Weibo, Gmail

IV. Experimental Design and Setup

A. Evaluation Metrics and Objectives

TSFN [2] performce evalated using standard classification metrics, including accuracy, precision, recall, and F1 score. Given the imbalanced nature of the traffic classes in the dataset, TSFN [2] focus on the macro-averaged F1 score is primary evaluation metric. The macro-averaged [6] F1 score ensures a balanced evaluation across all classes, regardless of their frequency, providing a more reliable measure of the model's ability to detect malicious traffic, especially in scenarios where some classes are underrepresented.

1. Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

TABLE II: The pre-processed data.

Label	Content
0	34d2 1a93 951a 4322 7c01 55ab 7b99 c512 3c9e 0a01 9805 7fd0 9830 8c70 58c0 2738 8bb9 4f33 42a9 2b10 b94b 72a0 a0e2 a9f3 90d1 689d 4f25 7856 51a7 f5cc 9d6a bcd3 09d1 58fe 9e13 5131 3a7d d825 04e1 b32b c9fc 8f57 68b4 4f0d d929 7bc1 8f8c a88b 3919 f0e5 9ac4 11b3 e00b 4120 5128 989f 9d92 99be d54e
1	a8f4 10e6 93c4 33c0 2f0d 71c1 4c93 5d0a 0101 3b79 b72d 9f92 59ea 7023 8b57 4011 2d62 07f7 2849 567c 9f51 d90a 34c2 3be5 9836 7d0b 9508 6c82 480f a4b1 5e90 01b9 d16b 5780 6c27 2837 4f2d 93e0 bdbd 6271 9e5c 3b0a 91fa 7035 9c0a 4a7d 853b c071 09e0 332b 3bfa 51ac 7284 6f50 9f92 3fbd 1048 7bc2
2	26a3 4f99 82b1 01f2 5397 4de2 82b1 d98d 7f4c 9d11 b0c2 920b d92f 84b9 5689 4001 8e36 87c1 0a8b 5301 6345 d0fc a0e9 4a81 4732 0a4a 0cc2 a2b7 91e7 78a0 c111 f61a 1013 b6fd 94f9 5093 529f a4b2 4103 8d56 0205 9c91 184e 8a5d 8b2e 97a3 759f a5b3 d7a2 5190 14b4 6a74 b7e3 d8fe 1a12 4108 90f9 3b87 18a5
3	56c3 a72d 41fe 3c79 1a84 2c4a b01a 9523 8a9f 56c1 7f38 33d0 9a82 a670 3f1a c4d2 748f 92cd a32b 5489 9173 7b2b 59d7 802d 24b1 d9fa 93d7 b7c3 d291 82b4 59c2 0713 68c7 52f3 054b 05a7 3d74 f473 53da 2e6f c233 6e5a 8e7c 3f3f 8712 09ad 9a34 59c9 46a0 88c4 7b7e 3d65 72fd a1c8

Accuracy is the ratio of the correctly thinked instances (true positives and true negatives) to the total instances. However, it might be misleading when the classes are imbalanced.

2. Precision:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

Precision, also known as positive predictive value, indicates the accuracy of positive predictions. It tells you how many of the items classified as positive are actually positive.

3. Recall (Sensitivity or True Positive Rate):

$$\text{Recall} = \frac{TP}{TP + FN} \tag{3}$$

Recall measures the ability of model to identify all relevant examples within the positive class. A high recall means fewer positive instances are missed.

4. F1-score:

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{4}$$

F1 score is what we call mean of precision and recall providing a balance between them. It is particularly useful when the classes are imbalanced.

B. Network Traffic Pre-processing Algorithm

The following algorithm outlines the preprocessing steps for transforming raw network traffic into a suitable format for input into the TSFN [2] model. The raw packet data is first transformed into hexadecimal string representations, then passed through a contextual embedding model (BERT) for packet-level feature extraction. The processed data is subsequently passed to the LSTM encoder for sequential analysis.

Algorithm 1: Network Traffic Pre-processing with BERT Embeddings

Input: PCAP files, batch size b, max packets M_p, sequence length l_s, max flows M_f
Output: Processed BERT embeddings tensor $\mathbf{E}$

Initialize BERT tokenizer and model
Initialize statistics tracker $\mathcal{S}$
foreach $pcap \in pcap\_files$ **do**
  foreach $mode \in \{flows, packets\}$ **do**
    $embeddings \leftarrow \emptyset$, $batch \leftarrow \emptyset$
    $flow\_set \leftarrow \emptyset$
    foreach $packet \in pcap$ **do**
      if *IsARP(packet)* $\vee$ *IsDHCP(packet)* **then**
        UpdateFilterStats($\mathcal{S}$)
        continue
      if *ValidatePacket(packet)* **then**
        $conn \leftarrow$ ExtractConnection($packet$)
        if $mode = flows \wedge conn \notin flow\_set \wedge |flow\_set| < M_f$ **then**
          $flow\_set \leftarrow flow\_set \cup \{conn\}$
          ProcessPacket($packet$, $batch$)
        else if $mode = packets \wedge |processed| < M_p$ **then**
          ProcessPacket($packet$, $batch$)
        if $|batch| \geq b$ **then**
          $hex\_data \leftarrow$ ConvertToHex($batch$)
          $tokens \leftarrow$ Tokenize(hex_data, l_s)
          $batch\_emb \leftarrow$ GenerateBERTEmbeddings($tokens$)
          $embeddings \leftarrow [embeddings; batch\_emb]$
          UpdateStats($\mathcal{S}$)
          ClearBatch($batch$)
        end
      end
    end
    ProcessFinalBatch($batch$, $embeddings$)
    SaveEmbeddings($embeddings$, $mode$)
    LogStatistics($\mathcal{S}$, $mode$)
  end
end

Function ValidatePacket($packet$):
  return HasIP($packet$) $\wedge$ (HasTCP($packet$) $\vee$ HasUDP($packet$)) $\wedge$ HasPayload($packet$)

Function ProcessPacket($packet$, $batch$):
  $filtered\_payload \leftarrow$ FilterPayload($packet$)
  $batch \leftarrow batch \cup \{filtered\_payload\}$

Algorithm 2: Noise-Robust LSTM Training for Network Traffic Classification

Input: $benign\_files$, $malware\_files$: Network traffic embeddings
$sequence\_lengths$: List of sequence lengths
$noise\_level$, $flip\_rate$: Robustness parameters
Output: Trained LSTM model and performance metrics

foreach $(label, files) \in \{(0, benign\_files), (1, malware\_files)\}$ **do**
  foreach $embeddings\_path \in files$ **do**
    $embeddings \leftarrow$ LoadEmbeddings($embeddings\_path$)
    $embeddings \leftarrow embeddings + \mathcal{N}(0, noise\_level)$
    $train, val, test \leftarrow$ SplitData($embeddings$, $labels$)
    ApplyLabelFlipping($train.labels$, $flip\_rate$)
    foreach $seq\_len \in sequence\_lengths$ **do**
      $model \leftarrow$ InitializeLSTM($embedding\_dim = 768$, $hidden\_dim = 64$)
      for $epoch \leftarrow 1$ **to** num_epochs **do**
        foreach $batch \in train$ **do**
          $output \leftarrow model(batch.data)$
          UpdateModel($model$, CrossEntropyLoss($output$, $batch.labels$))
        end
        $metrics_{val} \leftarrow$ EvaluateModel($model$, val)
        LogMetrics($epoch$, $metrics_{val}$)
      end
      $metrics_{test} \leftarrow$ EvaluateModel($model$, $test$)
      SaveResults($metrics_{test}$, seq_len)
    end
  end
end

Algorithm 3: LSTM Model Architecture

Input: $embedding\_dim$: Dimension of input embeddings
$hidden\_dim$: Dimension of LSTM hidden state
$output\_dim$: Dimension of output layer
Output: LSTM classifier model

`LSTMClassifier(`$embedding\_dim$, $hidden\_dim$, $output\_dim$`)` $dropout\_rate \leftarrow 0.6$
$lstm\_layers \leftarrow 1$
$lstm \leftarrow$ LSTM($embedding\_dim$, $hidden\_dim$, $lstm\_layers$)
$dropout \leftarrow$ Dropout($dropout\_rate$)
$fc \leftarrow$ LinearLayer($hidden\_dim$, $output\_dim$)
Function `Forward(`x`):`
  $lstm\_out, \_ \leftarrow lstm(x)$
  $final\_output \leftarrow lstm\_out[:, -1, :]$ `/* Take last sequence output */`
  $dropped \leftarrow dropout(final\_output)$
  return $fc(dropped)$

C. Description of Algorithms

Algorithm 1 focuses on converting packet data from PCAP files into BERT embeddings so that machine learning and downstream analysis of network data may be done efficiently. It controls the batching and processing of network packets according to predetermined limitations for packet counts, batch sizes, sequence lengths, and maximum flows. It functions primarily in two modes: flows and packets. The technique checks each packet in a PCAP file for IP, TCP/UDP headers, and the presence of the payload after filtering out unnecessary traffic types like ARP and DHCP. In the flows mode, packets are aggregated by connection, and in the packets mode, they are aggregated by total count. Valid packets are handled based on the mode. Upon reaching the batch size threshold, the algorithm tokenizes payloads, changes their format to hexadecimal, and generates embeddings using a BERT model. Processed embeddings are stored for each mode, while statistics are logged for performance tracking and evaluation.

Algorithm 2 describes a technique for training a noise-resistant LSTM model to identify network data as either malicious or benign. Initially, it loads network traffic embeddings and adds Gaussian noise to replicate real-world unpredictability. To improve robustness against label noise, a controlled label-flipping method is applied to a subset of the training labels after the embeddings have been divided into training, validation, and test sets. The technique initializes an LSTM model and trains it across several epochs, updating model weights using cross-entropy loss, for each given sequence length. The model is assessed on the validation set at each epoch, and performance is monitored by logging pertinent indicators. The model is evaluated on the test set following training in order to assess generalization, with final performance metrics saved for each sequence length. In order to increase the LSTM model's resistance to any data inconsistencies and facilitate efficient network traffic classification, this method makes use of noise addition and label flipping.

A classifier for sequential data, namely network traffic embeddings, is defined in Algorithm 3. The model has a fully connected output layer, a dropout layer (rate 0.6) to lessen overfitting, and a single-layer LSTM to capture temporal patterns. During the forward pass, the model applies dropout, utilizes the final output time step, processes the input sequence via the LSTM, and maps the outcome to the output layer for classification. The robust classification of network data and effective sequence representation are the goals of this architecture.

D. Experimental Configuration

The experiments were done on a system with a NVIDIA Tesla T4 GPU to ensure efficient handling of the computation-

ally intensive BERT-LSTM architecture. The dataset was split into training (80), validation (10), and test (10) sets, with data augmentation techniques used to ensure robust performance across different traffic classes. During the training process, we utilized the AdamW [5] optimization algorithm over 4 epochs. The hyperparameters included a learning rate of 2e-5, a batch size of 32 samples, and a dropout rate of 0.5. We evaluated the model on both packet-level and flow-level representations to assess its performance in various traffic scenarios. The model was trained over several epochs with early stopping criteria to prevent overfitting.

E. Hyperparameter Tuning

Hyperparameters such as batch size, learning rate, sequence length, and dropout rates were tuned through grid search to identify the optimal configuration for the TSFN [2] model. The learning rate was adjusted within a small range to ensure stable convergence, and dropout rates were carefully set to mitigate overfitting. Additionally, the model's performance was validated using cross-validation on the training set to ensure its robustness.

V. Results and Comparative Analysis

A. Baseline Comparisons

To evaluate the effectiveness of TSFN [2], its compared its performance against several baseline models, including both traditional machine learning models and recent deep learning approaches.

- **Traditional ML Models:** Models such as Random Forest, Support Vector Machine (SVM), and Decision Trees were trained on the same dataset. While these models achieved reasonable performance, TSFN [2] outperformed them by a significant margin, achieving a packet-level F1 score of 99.49. This is a notable improvement, demonstrating TSFN [2]'s ability to capture complex patterns and dependencies in encrypted traffic.
- **Deep Learning Models:** also compared TSFN [2] against deep learning architectures such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which have shown promise in traffic classification tasks. While CNNs and RNNs demonstrated improved performance over traditional ML models, they still fell short in handling encrypted traffic with high temporal dependencies. TSFN [2], by integrating BERT[5] for feature extraction and LSTM for sequential analysis, achieved superior performance in both packet-level and flow-level classification.

B. Analysis of appropriate sequence length and number of LSTM layers

Fig 3 depicts the impact of varying input sequence lengths on the performance of our model. The length of the input sequence plays a pivotal role in training BERT models, influencing both their efficiency and effectiveness. In this context, sequence length refers to the total number of tokens the model processes at once. If the length of the input sequence is shorter than the defined sequence length, padding is applied; conversely, if it exceeds the limit, truncation occurs. Longer sequences are more prone to incorporating redundant padding tokens [PAD], while shorter sequences might result in the omission of critical traffic data. Reducing the sequence length too much can lead to significant loss of important information. Therefore, it is crucial to carefully select the optimal sequence length to achieve the best performance. This section examines how different sequence lengths affect the model's accuracy. The results indicate that performance improves up to a certain point, before declining as sequence length continues to increase. The highest accuracy for detecting malicious traffic is observed when the sequence length is set to 160.

Fig 4 shows how changing the number of LSTM layers affects our model's performance. One important hyperparameter for LSTM models is the number of layers. By adding complexity and improving the model's capacity to fit the training data, more layers can increase the prediction accuracy of the model. By capturing data at various time steps, each extra LSTM layer helps the model learn and represent time series patterns more effectively. But if there are too many layers, the model might overfit the training set, which would make it perform worse on unobserved data. A model with too many layers and too much concentration on certain training set characteristics may lose its capacity to generalize.

	Precision	**Recall**	**F1-Score**	**Accuracy**
BitTorrent	96.5	97.2	96.8	99.0
Cridex	95.9	100.0	97.9	98.5
Facetime	100.0	97.9	98.9	99.1
FTP	96.7	96.4	99.5	100.0
Geodo	97.8	100.0	98.9	98.0
Gmail	100.0	96.2	98.0	96.0
Htbot	99.1	98.8	98.9	100.0
Miuref	97.4	97.9	97.6	97.8
MySQL	100.0	96.0	97.9	99.3
Neris	94.5	100.0	97.2	95.0
Nsis-ay	96.9	97.2	97.0	100.0
Outlook	100.0	97.5	98.7	97.8
Shifu	99.0	98.7	98.8	100.0
Skype	98.4	98.2	98.3	98.5
SMB	97.5	97.3	97.4	100.0
Tinba	100.0	96.0	99.0	96.2
Virut	94.9	100.0	97.4	99.4
Warcraft	96.8	97.1	99.0	97.2
Weibot	97.3	97.2	97.2	100.0
Zeus	98.1	100.0	99.0	98.4

TABLE III: Performance metrics for various classes.

VI. Discussion and Future Work

A. Advantages of TSFN [2]

TSFN [2] offers several key advantages over traditional and recent traffic classification models. One of the primary

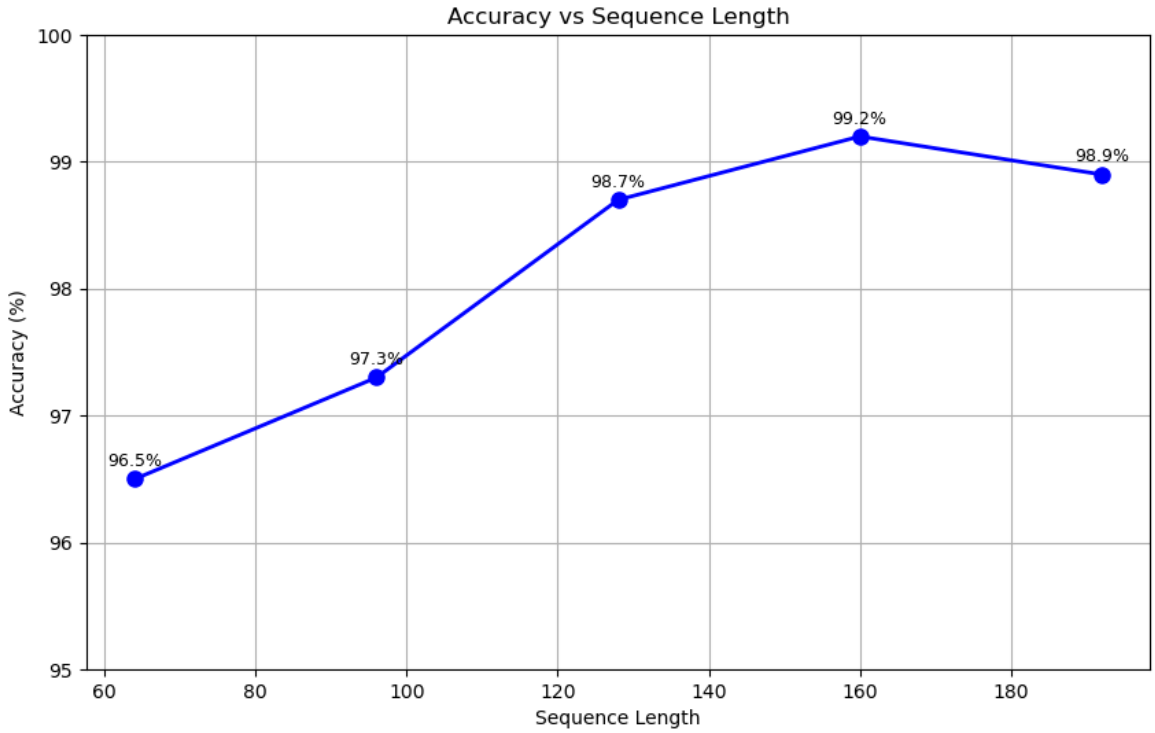

Fig. 3: Impact of varying sequence lengths on accuracy

benefits is its ability to effectively analyze encrypted traffic, where packet payloads are inaccessible. By combining the power of BERT for global feature extraction and LSTM for sequential pattern recognition, TSFN [2] provides a comprehensive solution for detecting malicious traffic in encrypted environments. The integration of BERT, particularly through the ET-BERT architecture, enables the model to capture intricate contextual relationships between packets, even in encrypted formats, where content is hidden. Moreover, the inclusion of tasks like the Masked BURST Model (MBM) and Same-origin BURST Prediction (SBP) further strengthens the model's capacity to identify dependencies at both the byte and burst levels, allowing TSFN to handle complex and varied attack scenarios effectively.

This dual-layered approach allows TSFN [2] to capture both static and dynamic features, making it highly adaptable to a wide range of attack scenarios. Additionally, the use of LSTM layers helps TSFN [2] recognize sequential patterns in network traffic, which is critical for detecting time-dependent threats. By processing both static flow characteristics and temporal relationships, TSFN [2] ensures a comprehensive understanding of network traffic.

Another advantage of TSFN [2] is its ability to generalize across different traffic types and attack patterns. Through extensive experimentation with the USTC-TFC dataset, we have shown that TSFN [2] performs well in detecting a variety of malware families and benign traffic types, making it a versatile tool for network security.

B. Limitations

Despite its strengths, TSFN [2] does have certain limitations. The most notable of these is its high computational cost. The use of both BERT and LSTM layers requires significant computational resources, especially when processing large volumes of network traffic in real-time. BERT's ability to model complex dependencies between packets and the additional BURST-related tasks introduce significant complexity, leading to increased processing time and memory requirements. As a result, further optimization of the model, particularly in terms of memory usage and inference speed, may be necessary for deployment in real-world applications. Additionally, while TSFN [2] shows impressive performance on the USTC-TFC dataset, it has yet to be tested on a wider range of datasets, which could reveal new challenges or limitations that need to be addressed.

C. Future Directions

In future work, several directions for enhancing TSFN [2] can be explored. One potential avenue is the integration of attention mechanisms within the model, which would allow TSFN [2] to dynamically focus on the most relevant parts of the traffic sequence. Attention mechanisms could further improve the model's ability to detect attacks, especially in scenarios

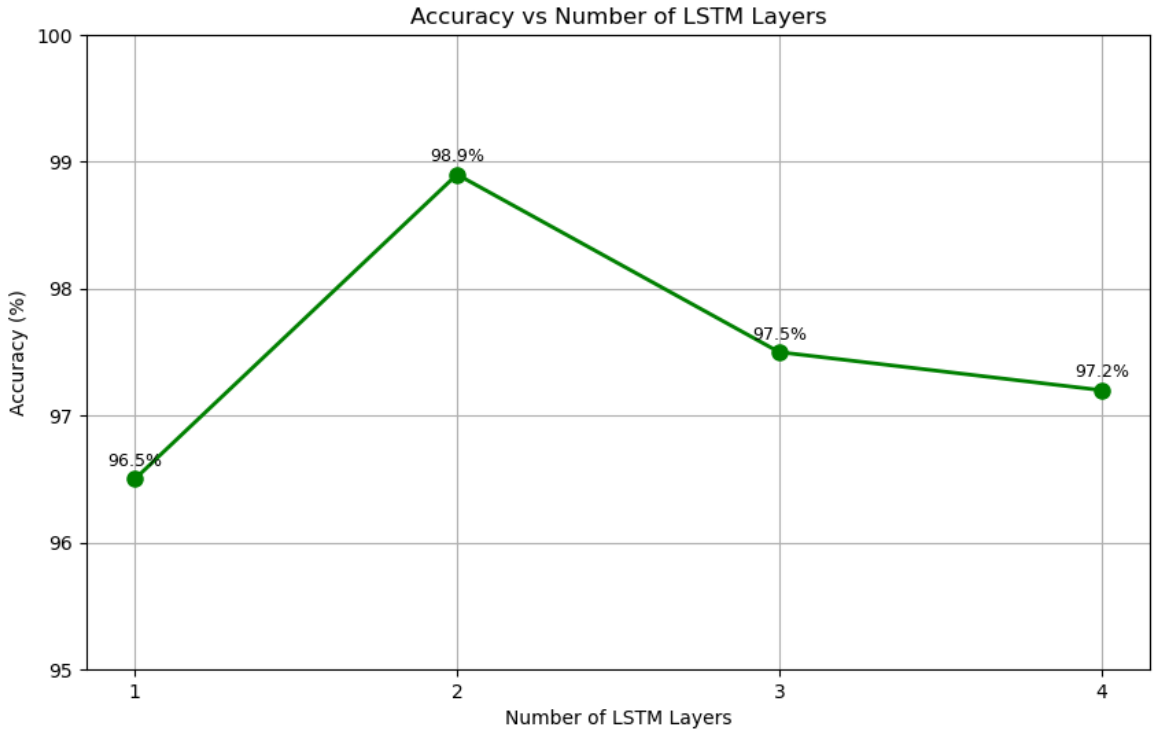

Fig. 4: Impact of varying LSTM layers on accuracy

where malicious activity is localized in specific traffic patterns. This could be particularly beneficial in encrypted traffic, where attack patterns may be subtle and difficult to distinguish without precise attention to contextual and structural features.

Another promising direction is the incorporation of unsupervised pre-training techniques. By pre-training the model on large, unlabeled datasets, TSFN [2] could potentially learn even more generalizable feature representations, improving its performance in detecting novel or unknown attack patterns. The combination of unsupervised learning with the Masked BURST Model (MBM) and Same-origin BURST Prediction (SBP) tasks could make TSFN [2] more adaptable to new, unseen encryption protocols or attack strategies. Additionally, exploring hybrid approaches that combine TSFN [2] with other pre-trained models, such as those based on reinforcement learning or graph-based models, could further enhance its adaptability and robustness to emerging threats in the ever-evolving landscape of network security.

VII. Conclusion

In this paper, we introduced the Time Series Feature Network (TSFN [2]), an advanced classification framework designed to address the increasing challenge of detecting malicious network traffic in encrypted environments. TSFN [2] integrates a BERT-based contextual embedding model with LSTM layers to capture both static and dynamic features within network flows. The BERT-based model, adapted for network traffic, leverages contextual relationships between packets, enabling the identification of complex traffic patterns even in encrypted formats. In particular, the Masked BURST Model (MBM) and Same-origin BURST Prediction (SBP) tasks, as part of the ET-BERT architecture, contribute to its ability to effectively handle encrypted traffic by learning dependencies within datagram sequences and identifying burst-level structural patterns, respectively. These innovations significantly enhance ET-BERT's accuracy in recognizing encrypted traffic patterns across diverse protocols.

Through extensive testing on the USTC-TFC dataset, the model demonstrated superior accuracy and robustness compared to baseline models, validating its effectiveness in real-world encrypted traffic scenarios. The combination of BERT's contextual embeddings, the BURST prediction tasks, and LSTM's sequential learning abilities allows TSFN [2] to capture both long-term dependencies and intricate packet-level relationships, making it a powerful tool for encrypted traffic classification.

The results highlight TSFN [2]'s potential in enhancing network security, particularly as encrypted communication becomes more prevalent.

Future work could focus on optimizing the model's efficiency and extending its adaptability to emerging network threats, further solidifying TSFN [2]'s applicability in evolving cyber defense landscapes. Additionally, efforts could be directed towards refining ET-BERT's ability to generalize across different

encryption protocols and improve its real-time performance in network monitoring.

For more information and access to the dataset used in this research, please visit the following link: https://github.com/yungshenglu/USTC-TFC2016

The complete source code for the data preprocessing pipeline and shape verification procedures can be accessed in the project's GitHub repository: https://github.com/thecypherops/replicate-tsfn

REFERENCES

[1] Lin, X., et al. "ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification." ACM Web Conference, 2022.

[2] Shi, Z., et al. "TSFN: A Novel Malicious Traffic Classification Method Using BERT and LSTM." Entropy, vol. 25, no. 821, 2023.

[3] Gao, M., et al. "Malicious Network Traffic Detection Based on Deep Neural Networks and Association Analysis." Sensors, vol. 20, no. 5, 2020.

[4] Ongun, T., et al. "On Designing Machine Learning Models for Malicious Network Traffic Classification." arXiv preprint, arXiv:1907.04846, 2019.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. 4171–4186.

[6] Chuan Liu, Wenyong Wang, Meng Wang, Fengmao Lv, and Martin Konan. 2017. An Efficient Instance Selection Algorithm to Reconstruct Training Set for Support

[7] Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. arXiv 2014, arXiv:1412.6980.

[8] Lin, Xinjie and Xiong, Gang and Gou, Gaopeng and Li, Zhen and Shi, Junzheng and Yu, Jing ET-BERT: A Contextualized Datagram Representation with Pre-training Transformers for Encrypted Traffic Classification