International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 11-No.3 (2025) pp. 4007-4018
http://www.ijcesen.com

ISSN: 2149-9144

Research Article

An Efficient Approach for Encrypted Traffic Classification and Intrusion Detection using Packet Transformer Encoder and CNN

K. Harinath [1] and G. Kishor Kumar [2*]

[1] Research Scholar, Department of CSE, JNTUA, Ananthapuramu, 515001, Andhra Pradesh, India.
Email: harirooba007@gmail.com – **ORCID**: 0000-0001-7854-5834

[2*] Department of CSE (AI & ML), Rajeev Gandhi Memorial College of Engineering and Technology, Nandyal, 518501, Andhra Pradesh, India.
* **Corresponding Author Email:** kishorgulla@yahoo.co.in – **ORCID:** 0000-0001-9624-2968

Article Info:

DOI: 10.22399/ijcesen.1377
Received : 20 January 2025
Accepted : 28 March 2025

Keywords :

Encrypted Traffic
Deep Learning
Classification
BERT
CNN

Abstract:

As encryption technology rapidly progresses and applications experience exponential growth, the research focus on network traffic classification has intensified. Current methods for classifying encrypted traffic exhibit certain constraints. Traditional techniques, including machine learning, heavily depend on feature engineering. Deep learning approaches are vulnerable to the quantity and distribution of labeled data, while pretrained models predominantly emphasize global traffic features, neglecting local features. In addressing these challenges, we introduced a methodology that incorporates both Bidirectional Encoder Representations from Transformers (BERT) and Convolution Neural Networks (CNN). To underscore both global traffic patterns and local features, we leverage the BERT and CNN mechanisms, respectively. Our approach attains state-of-the-art performance on the publicly accessible ISCX-VPN dataset for both traffic service and application identification tasks, achieving impressive F1 scores of 99.11% and 99.41%, respectively, in these domains. The experimental outcomes affirm that our method significantly enhances the performance of encrypted traffic classification.

1. Introduction

The classification of network traffic into distinct categories, known as traffic classification, holds significant importance in a variety of applications [1]. These applications include managing Quality of Service (QoS), pricing strategies, resource allocation planning, and the detection of malware and intrusions, etc. Recognizing its significance, a wide array of approaches has been devised to cater to the evolving and diverse requirements of various application scenarios. Notably, recent developments in communication technologies, encompassing encryption and port obfuscation, have introduced further complexities to the realm of network classification.

Traffic classification techniques have undergone substantial evolution over time [2]. The initial and most straightforward method involved using port numbers, although its accuracy has diminished due to newer applications employing well-known port numbers to conceal their traffic or bypassing standard registered port numbers. Despite its decreased accuracy, port numbers are still commonly used, either on their own or in combination with other attributes, in practical applications. The subsequent generation of traffic classifiers shifted their focus to packet content, utilizing data packet inspection (DPI) to identify patterns or keywords within data packets. These DPI methods are primarily effective for unencrypted traffic but come with a high computational overhead. The methods such as port numbers and DPI based for traffic classification are not capable enough for modern traffic circumstances towards encrypted traffic [3].

In light of the rapid advancement in internet technology and the growing emphasis on data privacy, contemporary network application traffic commonly undergo encryption using a range of encryption protocols to guarantee data security and privacy [4,5]. Leading the pack among these encryption protocols are Secure Socket Layer (SSL) and its successor, Transport Layer Security

(TLS) [7,40]. According to a recent report from Google, as of October 2021, 98% of web pages loaded on Chrome have implemented SSL/TLS encryption.

The SSL/TLS communication process comprises two main stages: the handshake and the formal data transmission. During the initial handshake, the client initiates the process by sending a Client Hello to the server, encompassing a random number and the supported cipher suites. The server responds with a Server Hello, containing the agreed-upon cipher suite, server certificate, and another random number. Subsequently, both parties collaboratively derive a final master key using the exchanged information. The server then concludes the handshake by sending a Server Finish to the client. In the second stage, utilizing the established master key, both parties employ symmetrical encryption for data transmission, enhancing decryption speed compared to asymmetric encryption methods.

The primary difficulty in classifying encrypted traffic lies in effectively characterizing the encrypted data stream. In contrast to plaintext network traffic, which can be identified through deep packet inspection, encrypted packets defy classification based on their content alone. While the Server Name Indication (SNI) field in TLS handshake packets can occasionally offer insights into the nature of the traffic, it's worth noting that not all cases include complete handshake packets. An encouraging approach to tackle encrypted traffic classification involves the application of machine learning algorithms [9-11]. Nevertheless, the majority of these techniques rely on manually crafted flow features, leading to the omission of crucial packet details and hindering the possibility of fine-grained operations. Consequently, this compromises the accuracy of classification.

Addressing this challenge involves the comprehensive amalgamation of information from various components within TLS traffic to achieve a more precise classification. When examining network packets, one must consider the IP, TCP, TLS headers, payload, and other elements. Simultaneously, network flows comprise handshake, transmission, and communication finish packets. In terms of informational content, network traffic encompasses flow details, statistical data, packet lengths, TLS message types, and more. Numerous prior studies have focused on the integration of diverse information sources, such as incorporating certificate packet lengths into hand-designed features [12] [13] or appending TLS message types to packet length sequences [14]. Despite these efforts, effective methods for integrating a diverse array of information sources remain elusive to date [15].

To enhance the classification accuracy of encrypted traffic, preserve information integrity to the fullest extent, and introduce finer packet-level operations, this paper introduces a model that incorporates both transformer and convolutional neural network (CNN) architectures. The rationale behind employing the multi-head self-attention mechanism in transformers is to concentrate on the data content from various viewpoints, aligning with the aforementioned challenges. Further the CNN is used for classifying the network traffic into various categories.

The rest of the paper is organized as follows. The related work on encrypted traffic classification is given in Section 2. Section 3 gives the problem definition and also the proposed work. Section 4 discusses the datasets used for various experiments. Section 5 gives the experimental analysis on the datasets used for encrypted traffic classification. The paper is concluded in Section 6.

2. Literature Survey

This section provides an overview of relevant survey literature encompassing diverse encrypted traffic classification methods. The review covers traditional approaches, methods based on message types, length sequences, message types, statistics, and end-to-end techniques for traffic classification. The Conventional approaches comprise port-based methods [16] and payload-based methods [17]. Port-based methods rely on identifying applications through a port registration list provided by the Internet Assigned Numbers Authority (IANA). However, this method's reliability diminishes as more applications utilize dynamically allocated ports or employ common communication protocol ports for camouflage. The payload-based method, also known as deep packet detection (DPI), classifies applications by detecting key strings in network traffic. The work presented in [18] employed application-level signatures to classify P2P application traffic, while the research in [8] utilized statistical application signatures. Unfortunately, both methods are limited to handling unencrypted network traffic and prove entirely ineffective for encrypted traffic.

Length-based methods abstract network flows into sequences of lengths, employing Markov models or other machine learning techniques to characterize these sequences. The rationale behind utilizing length sequences as packet representations lies in the significant variations in packet size patterns among different application traffics. For instance, upload/download type traffic typically exhibits larger average packet sizes compared to chat-type traffic. An example of such an approach is App

scanner, as introduced in [19] and [20], which leverages a packet length vector to classify mobile applications. Employing the Random Forest algorithm, it successfully categorizes the 110 most popular bot-generated Android applications, achieving an impressive application re-identification accuracy of up to 96% in optimal conditions. Recently, in [53], they proposed FS-Net, a framework for classifying encrypted traffic using deep neural networks. FS-Net employs packet length sequences as input, utilizes bidirectional GRU [22] for feature encoding, and introduces a reconstruction mechanism in AutoEncoder to ensure the validity of the learned features. While length sequences have demonstrated effectiveness in encrypted traffic classification across various prior works, relying solely on packet length for representation is a simplistic approach that may overlook important details. In cases where packet lengths are similar or identical (e.g., packet fragmentation in the IP layer), the length sequence loses its discriminative power.

The SSL/TLS packet headers incorporate a field identifying the message type within each packet. Initially, Korczyriski et al. [23] introduced a unidirectional firstorder homogeneous Markov model for encrypted traffic classification, based on the sequence of message types. Subsequently, Shen et al. [13] proposed a second-order Markov model that integrates the certificate packet length with the length of the initial communication packet. However, this approach is not without limitations: i) the Markov model relies on a small number of data points (two or three-time steps) for training, resulting in a lack of comprehensive sequential information; ii) the limited number of message types leads to overlapping issues [15]. Some studies have attempted to address these challenges by combining length sequences and message types [15], offering a partial solution to the representational limitations of a single message type feature. Beyond message types, several other fields, including cipher suite, version, and certificate information, can convey information about encrypted HTTPS traffic. Nonetheless, these methods are specific to SSL/TLS traffic and are not applicable to a broad spectrum of protocols, including unencrypted HTTP, RTP, SRTP, MTProto, and various general mobile application classifications.

Statistical feature-based methods in network traffic analysis aim to derive flow level statistical features, subsequently employing machine learning algorithms for classification. These features typically like average packet length, average time interval, and transmission rate. Open-source tools like CICFlowMeter [24] and Joy [25] facilitate the extraction of such features. Zhu et al. [26] introduced the Attention based Multi-Flow LSTM (AMF-LSTM) model for network attack detection, leveraging multiple flows to incorporate historical network information. Their approach utilized Attention Mechanism (AM) to identify network traffic with significant contributions to classification. Shbair et al. [27] proposed a two-level hierarchical framework for traffic classification. This framework also introduced a new set of statistical features for categorizing services operating over HTTPS connections. Evaluation results indicated that 50 out of the 68 considered HTTPS services achieved a recall rate above 95%. Despite their effectiveness, these approaches have limitations, including highly abstracted features that hinder fine-grained operations (such as learning the relationship between two packets). Additionally, the extraction of statistical features often requires observing the entire network flow until its completion, rendering real-time traffic classification impractical.

Recent efforts have explored strategies for end-to-end encrypted traffic classification, where" end-to-end" refers to the direct utilization of raw network packet bytes without manually designed features. This approach maximizes the potential of neural networks to autonomously discover hidden features. Currently, Convolutional Neural Networks (CNN) serve as the primary method for end-to-end traffic classification [28], Wang et al. [29] proposed a technique involving the conversion of packets into images, processed with 1D-CNN, achieving notable performance on the ISCX VPNNon-VPN traffic dataset. This dataset was also assessed in [30], employing Stacked Autoencoders (SAE) and CNN. The approach retained the IP header and the initial 1480 bytes of each IP packet as input, achieving over 90% accuracy in classifying 17 applications at the packet level. However, pure CNN exhibits limitations in representing network flow characteristics as it cannot capture interactive information across different time steps. Addressing this, [31] and [32] initially employed CNN to learn features from the first 784 bytes of each packet, then combined these features using Long Short-Term Memory (LSTM) to obtain the flow feature vector. Rezaei et al. [33] adopted a similar strategy, learning the header and payload of the first six packets and achieving effective identification of ambiguous flows. Despite the integration of more complex joint models, these works fundamentally represent single-modal end-to-end learning frameworks with inherent challenges related to weak feature representation.

3. Methodology

A network packet comprises multiple bytes, and the aggregation of packets occurring within a defined time frame forms the network traffic set denoted as S. Let S can be defined as follows:

$$S = \{P^1, P^2, \ldots, P^m\},\ m = |S| \qquad (1)$$

Each packet P^i , can be defined as

$$P^i = \{X^i, B^i, T^i\}, \qquad 1 < i < m;$$

$$X^i = \{SrcIP^i, DstIP^i, SrcPort^i, DstPort^i, Protcol^i\}.$$

$$B^i = \{byte^1, byte^2, \ldots, byte^p\},$$

$$0x00 <= byte^k <= 0xff, 1 <= k <= p.$$

$$T^i > 0;$$

Therefore, a packet P^i can be represented with a 5-tuple X^i, byte content B^i and the start time T^i. The collection of packets with the same X can be called as Flow, the k-th flow can be defined as follows (2).

$$f_k = \{P_k^1, P_k^2, \ldots P_k^l\}, \qquad l = |f_k| \qquad (2)$$

In (2),

$$X_k^1 = \cdots = X_k^l \quad and \quad T_k^1 < \cdots < T_k^l$$

Hence, the original traffic set can also be represented as,

$$R' = \{f1, f2, \ldots fk\}, k = |R'| \qquad (3)$$

For the given observed packet sequence *R* and the presence of *N* applications, our objective is to develop a model ϕ (f_l) that can predict the label L_l of a network flow f_l, $1 <= L_l <= N$.

3.1 BERT:

The introduction of the BERT model by the Google AI team marked a significant milestone in the field of Natural Language Processing (NLP), propelling NLP research forward dramatically [41]. BERT demonstrated outstanding performance in the Stanford Question Answering Dataset 1.1 (SQuAD) challenge [41], a prominent machine reading comprehension competition. It surpassed human performance across the board in two evaluation metrics and exhibited excellence in eleven different NLP tasks. BERT not only ushered in a new era of NLP but also gained widespread popularity in various other domains. Its effectiveness extended to the field of cyber-security, where it demonstrated exceptional performance.

3.2 Convolutional Neural Network:

The design of convolutional networks draws inspiration from the biological structure of vision [42], making convolutional neural networks (CNNs) particularly adept at learning local features or features with non-stationary locations [1]. CNNs have demonstrated exceptional performance across diverse domains, including natural language processing [43] and machine vision [44]. In recent years, CNNs have also exhibited remarkable proficiency in handling encrypted traffic. For instance, in the utilization of DeepPacket, as highlighted in [45], CNNs achieved a notably high accuracy rate on the ISCX dataset.

In image classification tasks, successive convolutional layers of CNNs progressively extract features from individual pictures. Notably, features extracted in the initial convolutional layers tend to be relatively straightforward, such as edges and curves. As the process advances to deeper convolutional layers, the extracted features become more intricate. The features obtained in the intermediate layers of the network often prove to be complex and challenging for humans to interpret [57]-[59].

3.3 Fully Connected Layer:

In deep learning models, the fully connected layer (FC) typically functions as a classifier, translating the data features acquired by the model into the labeled space. The fundamental operation of the fully connected layer involves a matrix–vector product, expressed by the following formula:

$$Y = W \times X + B \qquad (4)$$

In (4), W represents the weight of the fully connected layer, while *B* denotes the bias of the fully connected layer. The input and output vectors are represented by *X* and *Y*, respectively.

The BERT attention mechanism is employed to capture packet-level feature vectors, which are subsequently transformed into byte feature input vectors through a convolutional layer. This process generates locally salient features. Following the convolutional operations, the enriched feature vector, containing substantial information, serves as the input for a fully connected layer tasked with traffic prediction.

In the proposed framework, an encoder is employed to transform a series of symbolic representations

(x_1,...x_n) into a corresponding sequence of continuous representations denoted as z = (z_1,...z_n). Subsequently, utilizing the obtained continuous representation z, a decoder is responsible for generating an output sequence (y_1,...y_m) of symbols in a step-by-step fashion. The model operates in an autoregressive manner, as detailed in reference [34]-[35], wherein the generation of each symbol at a given step involves the incorporation of previously generated symbols as supplementary input. The Transformer adopts the overall structure by employing stacked self-attention and point-wise, fully connected layers in both the encoder and decoder. These components are illustrated in the left and right halves of Figure 2, respectively.

The transformer model architecture consists of various components such as encoder, decoder, attention mechanism etc. and each of these is discussed as follows. The encoder consists of a stack comprising $N = 6$ identical layers, each containing two sub-layers. The initial sub-layer is a multi-head self-attention mechanism, while the second is a straightforward, position-wise fully connected feed-forward network. A residual connection [11] encircles each of these sub-layers, followed by layer normalization [36]. In essence, the output of each sub-layer is obtained through the expression LayerNorm(x + Sublayer(x)), where Sublayer(x) represents the function executed by the sub-layer itself. To accommodate these residual connections, all sub-layers within the model, including the embedding layers, generate outputs of dimension d_{model} = 512. The decoder is structured with a stack of $N=6$ identical layers. While each encoder layer comprises two sub-layers, the decoder introduces a third sub-layer. This additional sub-layer conducts multi-head attention over the output of the encoder stack. Similar to the encoder configuration, residual connections surround each sub-layer, followed by layer normalization. Notably, the self-attention sub-layer in the decoder stack to prevent positions from attending to subsequent positions. This masking, coupled with the offset of one position in the output embeddings, ensures that predictions for position i are solely influenced by the known outputs at positions preceding i.

An attention function can be defined as a mechanism that takes a query and a collection of key-value pairs as input, producing an output. In this context, the query, keys, values, and output are all vector representations. The resulting output is determined through a weighted summation of the values, with each value's weight determined by a compatibility function that evaluates the relationship between the query and its corresponding key.

For Scaled Dot-Product Attention, the input comprises queries and keys, both of dimension d_k, and values with dimension d_v. The computation involves obtaining dot products between the query and all keys, dividing each result by d_k, and applying a softmax function to derive weights assigned to the values. In practical applications, the attention function is concurrently computed for a set of queries grouped into a matrix Q. Correspondingly, the keys and values are organized into matrices K and V. The resultant matrix of outputs is calculated as follows (5).

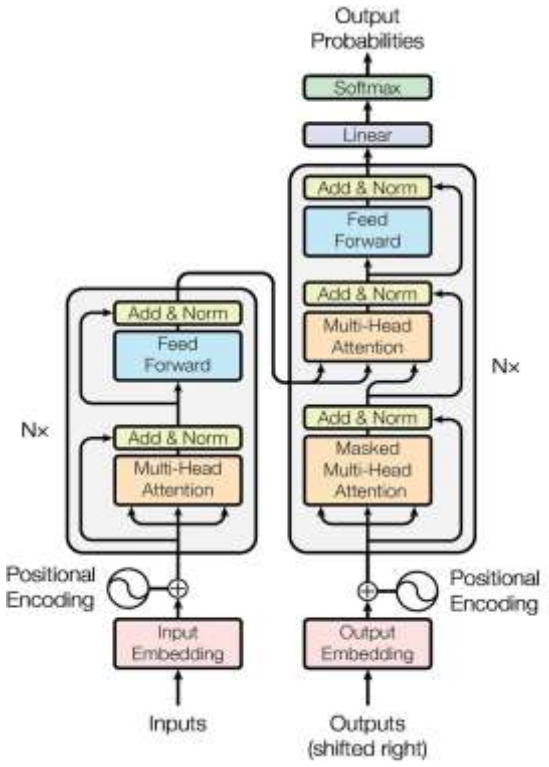

Figure. 1 *Transformer-Model-Architecture.*

$$Attention(Q,K,V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \qquad (5)$$

Two widely employed attention functions are additive attention [37] and dot-product (multiplicative) attention. Additive attention calculates the compatibility function using a feed-forward network featuring a single hidden layer. Although they share similar theoretical complexity, in practical terms, dot-product attention proves to be considerably faster and more space-efficient. This efficiency arises from its implementation utilizing highly optimized matrix multiplication code. Notably, while the two mechanisms exhibit comparable performance for small values of dk, additive attention surpasses dot-product attention without scaling for larger values of dk [38].

For the Multi-Head attention function, rather than executing a singular attention function with d_{model}-dimensional keys, values, and queries, we have observed advantages in linearly projecting the queries, keys, and values h times. Each projection involves distinct learned linear transformations, resulting in dimensions d_k, d_k and d_v, respectively. The attention function is then applied independently to these projected versions of queries, keys, and values in parallel, producing output values of dimension d_v. These outputs are concatenated and subjected to an additional projection, ultimately yielding the final values. The use of multi-head attention enables the model to concurrently focus on information from various representation subspaces at different positions. A single attention head, on the other hand, hinders this capability by averaging the information.

$$MultiHead(Q,K,V) = Concat(head_1,\dots,head_h)W^O \quad (6)$$

In (6),

$$head_i = Attention(QW_i^Q, KW_i^k, VW_i^V) \quad (7)$$

In (7), the projections are parameter matrices $W_i^Q \in \mathbb{R}^{d_{model}\times d_k}, W_i^K \in \mathbb{R}^{d_{model}\times d_k}, W_i^V \in \mathbb{R}^{d_{model}\times d_v} and\ W^O \in \mathbb{R}^{hd_v\times d_{model}}$

In this work we employ $h = 6$ parallel attention layers, or heads. For each of these we use $d_k = dv = d_{model}/h = 64$. Due to the reduced dimension of each head, the total computational cost is similar to that of single-head attention with full dimensionality.

4. Datasets

To assess the efficacy of our approach, we conducted experiments utilizing two datasets such as the ISCX VPN-nonVPN traffic dataset (ISCX-VPN) [39] provided by the University of New Brunswick and ISCX 2012 IDS dataset [56]. The first dataset comprises raw traffic data generated by various applications, each labeled based on the actions performed during the traffic capture (e.g., chats, file transfers, video calls) and the originating application (e.g., Youtube, Facebook, Netflix). Our experiments focused on service and application identification tasks using this dataset. The service categories encompassed 12 distinct categories, while the applications were classified into 17 categories, as detailed in Table 1.

To streamline the raw traffic data and adapt it to our model's input format, we initially employed the Datagram2Token tool [40]. This involved segmenting the dataset at the packet level and removing dynamic host configuration protocol (DHCP) and address resolution protocol (ARP) packets lacking pertinent information about the originating applications. To address potential bias inferences stemming from

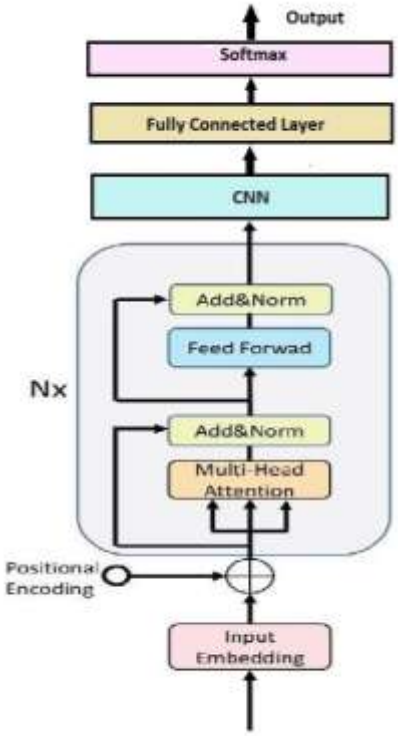

Figure. 2 *Proposed Model Architecture.*

robust discriminative information in packet headers, such as port numbers and IP addresses, we excluded the protocol ports of the TCP header, IP header, and Ethernet header. Subsequently, we utilized a bi-gram model to encode the resulting hexadecimal sequences [40]. To manage data volume, we randomly selected a maximum of 5000 packets from each class, maintaining a uniform sample size across all classes, except for the AIM-Chat and ICQ classes in the application identification task, where sample sizes were 1340 and 823, respectively. The dataset, aligned with ET-BERT [40], was then divided into training, validation, and test datasets in an 8:1:1 ratio. Our preprocessing tailored the ISCX-VPN dataset to suit the characteristics of both service and application identification tasks, leading to the creation of distinct ISCX-VPN-Service and ISCX-VPN-App datasets. The statistical information of the datasets is given in Table 2.

The second dataset utilized for assessing the efficacy of the proposed model in intrusion detection is derived from the ISCX 2012 IDS dataset [56]. This dataset encompasses seven days of network traffic, categorized into five classes:

Normal, Brute Force SSH, DDoS, HttpDoS, and Infiltrating. For simplicity, we exclusively focus on days with malware traffic. As malware traffic typically exhibits a smaller scale compared to normal traffic in real-world scenarios, we opt not to apply any normalization.

Table 1. *Details of the Dataset*

Dataset	Service	Application
ISCX-VPN	Chat	AIM-Chat
	Email	Email
	File Transfer	Facebook
	P2P	Gmail
	Streaming	Hangout
	VoIP	ICQ
	VPN-Chat	Netflix
	VPN-Email	SCP
	VPN-FT	Skype
	VPN-P2P	Spotify
	VPN-Streaming	Tor
	VPN-VOIP	Torrent
		Vimeo
		VoipBuster
		VPN-Ftps
		VPN-Sftp
		Youtube

techniques to the dataset. Table 3 illustrates the structure of the chosen dataset. Subsequently, this selected dataset is partitioned into training and testing sets in a 9:1 ratio for each class.

Table 2. *Statistical Details of the Dataset1.*

Dataset	No of Samples	No of Labels
ISCX-VPN-Service	60,000	12
ISCX-VPN-App	77,163	17

Table 3. *Statistical Details of the Dataset2: ISCX 2012 IDS*

Dataset	Class Names	No of Samples
ISCX 2012 IDS	Normal	848306
	Brute Force SSH	6964
	DDoS	22121
	HttpDoS	3482
	Infiltrating Transfer	10044

5. Experiments

To enhance the performance of encrypted traffic classification, we introduced a methodology integrating BERT and CNN. The evaluation of this model utilized the ISCX-VPN dataset, emphasizing traffic service and application identification tasks. This section presents the experimental results.

The experiments were conducted on the Ubuntu 20.04 operating system, employing a 14-core Intel® Xeon® Gold 6330 CPU @ 2.00 GHz as the processor and a single NVIDIA 3090 graphics processing unit with a 24 GB memory size. Throughout all experiments, PyTorch version 1.11.0 and universal encoder representations [46] were utilized. Model parameters included a maximum input length of 128 tokens, a batch size of 32, 16 epochs for training, a learning rate set at 2×10^{-5}, and the choice of the AdamW [47] optimizer.

We employed four evaluation metrics: accuracy (AC), precision (PR), recall (RC), and F1 score (F1). During model evaluation, the category of interest typically served as the positive class, with other categories considered as negative classes. True Positives *(TP)* represented the number of correctly identified positive samples, False Positives *(FP)* denoted the number of incorrectly identified negative samples, True Negatives *(TN)* indicated the number of correctly identified negative samples, and False Negatives *(FN)* referred to the number of wrongly identified positive samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \quad (8)$$

$$Precision = \frac{TP}{TP + FP} \quad (9)$$

$$Recall = \frac{TP}{TP + TN} \quad (10)$$

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (11)$$

The sequence length stands as a crucial parameter in the BERT model, allowing for sequences up to a maximum of 512. In cases where it exceeds this predefined length, the BERT model undergoes truncation, while shorter sequences undergo a fill token operation. The choice of sequence length significantly impacts model performance. Excessive values may introduce nonsensical tokens, while overly short lengths risk information loss in traffic data. Hence, a judicious selection of sequence length is imperative. Table 4 presents our model's accuracy across different sequence lengths.

Table 4. *Impact of Sequence Length in terms of Accuracy.*

Sequence Length	Service Identification	App Identification
8	68.24	92.91

16	98.89	99.52
32	99.12	99.14
64	99.01	99.14
128	99.28	99.76

Notably, an increase in sequence length generally led to improved overall accuracy, peaking at a sequence length of 128. Consequently, we opted to set the maximum

Table 5. *Impact of No of CNN Layers in terms of Accuracy.*

Number of CNN Layers	Service Identification	App Identification
1	98.82	99.63
2	99.12	99.69
3	99.39	99.76
4	98.96	99.73

sequence length at 128 for optimal model performance. In Table 5, the model's performance is displayed across different numbers of CNN layers for both service and application identification. With an increase in the number of layers, there was a corresponding rise in model accuracy. Optimal performance is achieved when utilizing three layers. However, beyond three layers, the model's performance either decreased or remained unchanged. This is attributed to the introduction of excessive parameters and computations, negatively impacting the model. Consequently, the decision was made to set the number of CNN layers in our model to three for an optimal balance between accuracy and computational efficiency.

6. Comparison Study

This paper selected 12 widely acknowledged methods in encrypted traffic classification as benchmark approaches. These methods utilize data that aligns with the data employed in our own methodology are: Fingerprint construction approach: FlowPrint [48]; Statistical feature approaches: AppScanner [50], CUMUL [49], BIND [51], and k-fingerprinting (K-fp) [54]; Deep learning approaches: deep fingerprinting (DF) [52], FS-Net [53], GraphDApp [54] and DeepPacket [45], CNN-TLBO[60];

Table 6. *Comparison results on service identification tasks.*

Method	Accuracy	Precision	Recall	F1-Score
AppScanner	71.82	73.39	72.25	71.97
CUMUL	56.1	58.83	56.76	56.68
BIND	75.34	75.83	74.88	74.2
K-fp	64.3	64.92	64.17	63.95
FlowPrint	79.62	80.42	78.12	78.2
DF	71.54	71.92	71.04	71.02
FS-Net	72.05	75.02	72.38	71.31
GraphDApp	59.77	60.45	62.2	60.36
DeepPacket	93.29	93.77	93.06	93.21
CNN-TLBO	97.00			
Proposed	99.28	99.14	99.13	99.14

Tables 6 and 7 present the experimental outcomes for our proposed model and other models in service identification and application identification using the ISCXVPN dataset. For the service identification task, our approach demonstrated superior performance across four evaluation metrics, outperforming all baseline models. In the

Table 7. *Comparison results on application identification tasks.*

Method	Accuracy	Precision	Recall	F1-Score
AppScanner	62.66	48.64	51.98	49.35
CUMUL	53.65	41.29	45.35	42.36
BIND	67.67	51.52	51.53	49.45
K-fp	60.7	54.78	54.3	53.03
FlowPrint	87.67	66.97	66.51	65.31
DF	61.16	57.06	47.52	47.99
FS-Net	66.47	48.19	48.48	47.37
GraphDApp	63.28	59	54.72	55.58
CNN-TLBO	97.00			
DeepPacket	97.58	97.85	97.45	97.65
Proposed	99.76	99.48	99.59	99.52

application identification task, our method excelled by surpassing baseline models in accuracy, recall, and F1 values, ultimately achieving the top performance.

Table 8 and Table 9 shows the performance of the proposed method for each type of encrypted traffic service identification and also for encrypted traffic application identification respectively. Upon thorough observation and analysis, we noted that relying solely on the CNN model yielded suboptimal performance. This inadequacy stems from CNN's efficiency in capturing local features while neglecting global features. In contrast, the BERT model, equipped with a multi-head attention mechanism, excels at extracting global features. By leveraging the strengths of both models, we enhance the overall performance of encrypted traffic classification.

Table 8. *Performance for each type of encrypted traffic service identification.*

Class	Precision	Recall	F1-Score

Chat	98.2	99.1	98.6
Email	99.1	97.9	98.5
File-Transfer	100	99.6	99.8
P2P	99.9	100	99.9
Streaming	99.9	99.8	99.8
VoIP	99.6	99.8	99.6
VPN-Chat	99.6	99.4	99.6
VPN-Email	99.6	100	99.8
VPN-FT	99.5	97.4	98.4
VPN-P2P	99.9	100	99.9
VPN-Streaming	99.9	99.8	99.8
VPN-VoIP	97.4	99.2	98.6

7. Conclusion

In this paper, we introduced a novel model for encrypted traffic classification, leveraging both BERT and CNN. Our model utilizes the BERT approach to extract global features and incorporates the CNN model to capture local features. The resulting data is then input into the classifier for predicting the corresponding class of the encrypted traffic. Experimental results demonstrate that our model attains state-of-the-art performance in service identification and encrypted traffic application recognition tasks on the ISCX-VPN dataset, achieving remarkable accuracy rates of 99.12% and 99.65%, respectively. While our approach exhibits superior performance, it is not without limitations. Our focus on VPN encrypted traffic excludes considerations for the characteristics of other encrypted traffic and the attributes of malicious traffic. Future research will extend our method to encompass a broader range of more traffic types.

Table 9. *Performance for each type of encrypted traffic application identification*

Class	Precision	Recall	F1-Score
AIM Chat	99.6	98.6	99.2
Email	99.6	100	99.8
Facebook	99.5	99.2	99.1
Gmail	99.1	99.4	99.1
Hangout	100	99.6	99.8
ICQ	95.6	97.9	97.2
Netflix	100	100	100
SCP	100	99.6	99.7
Skype	99.6	99.8	99.6
Spotify	100	100	100
Tor	100	100	100
Torrent	99.2	99.9	99.5
Vimeo	100	100	100
VoipBuster	100	98.9	99.6
VPN-Ftps	100	100	100
VPN-Sftp	100	99.8	99.9
Youtube	100	100	100

Table 10. *Performance for each type of intrusion detection*

Class	Precision	Recall	F1-Score
BFSSH	99.92	99.84	99.18
DDoS	98.74	98.96	98.96
HttpDoS	98.59	99.81	99.12
Infiltrating	98.71	98.84	98.16

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Zhang, Z., Han, X., Liu, Z., Jiang, X., Sun, M., & Liu, Q. (2019). *ERNIE: Enhanced language representation with informative entities* (arXiv:1905.07129). arXiv. https://arxiv.org/abs/1905.07129

[2] Rezaei, S., & Liu, X. (2019). Deep learning for encrypted traffic classification: An overview. *IEEE Communications Magazine*, 57(5), 76–81. https://doi.org/10.1109/MCOM.2019.1800819

[3] Zeng, Y., Li, K., He, X., Liu, X., & He, X. (2019). Deep-Full-Range: A deep learning-based network encrypted traffic classification and intrusion detection framework. *IEEE Access*, 7, 45182–45190. https://doi.org/10.1109/ACCESS.2019.2908430

[4] Lin, P., Ye, K., Xu, C.-Z., Wang, Y., & Li, J. (2021). PEAN: A packet-level end-to-end attentive network for encrypted traffic identification. In *2021 IEEE 23rd Int. Conf. on High Performance Computing & Communications (HPCC/DSS/SmartCity/DependSys)* (pp. 1183–1190). IEEE. https://doi.org/10.1109/HPCC-DSS-SmartCity-DependSys52092.2021.00161

[5] Liu, J., Duan, H., Li, B., He, Q., & Zhou, W. (2017). Effective and real-time in-app activity

analysis in encrypted internet traffic streams. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 2015–2024). ACM. https://doi.org/10.1145/3097983.3097992

[6] Lin, P., Ye, K., Xu, C.-Z., & Yang, J. (2022). A novel multimodal deep learning framework for encrypted traffic classification. *IEEE/ACM Transactions on Networking*, 30(2), 536–549. https://doi.org/10.1109/TNET.2021.3134495

[7] Sheela, M., Amirthayogam, G., Hephzipah, J. J., Suganthi, R., Karthikeyan, T., & Gopianand, M. (2024). Advanced brain tumor classification using DEEPBELEIF-CNN method. *Babylonian Journal of Machine Learning*, 2024, 89–101. https://doi.org/10.58496/BJML/2024/009

[8] Roughan, M., Sen, S., Spatscheck, O., & Duffield, N. (2004). Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification. In *Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement* (pp. 135–148). ACM. https://doi.org/10.1145/1028788.1028805

[9] Lin, P., Ye, K., & Xu, C.-Z. (2019). Dynamic network anomaly detection system by using deep learning techniques. In *Cloud Computing – CLOUD 2019: 12th International Conference* (pp. 243–252). Springer. https://doi.org/10.1007/978-3-030-23255-9_19

[10] Alazawi, S. A. H., Abdulbaqi, H. A., & Ali, A. H. (2024). CNN-based intrusion detection software for network operating system environment. *Babylonian Journal of Internet of Things (BJIoT)*, 2024, 79–86. https://doi.org/10.58496/BJIoT/2024/010

[11] Anderson, B., & McGrew, D. (2016). Identifying encrypted malware traffic with contextual flow data. In *Proceedings of the 2016 ACM Workshop on Artificial Intelligence and Security* (pp. 35–46). ACM. https://doi.org/10.1145/2996429.2996435

[12] Kalnoor, G., Sai, K., Dasari, S. S., Waddenkery, N., & Pragathi, B. (2024). Enhanced brain tumor detection from MRI scans using frequency domain features and hybrid machine learning models. *Journal of Modern Technology*, 2024, 141–149.

[13] Shen, M., Wei, M., Zhu, L., & Wang, M. (2017). Classification of encrypted traffic with second-order Markov chains and application attribute bigrams. *IEEE Transactions on Information Forensics and Security*, 12(8), 1830–1843. https://doi.org/10.1109/TIFS.2017.2687799

[14] Liu, C., Cao, Z., Xiong, G., Gou, G., Yiu, S.-M., & He, L. (2018). MaMPF: Encrypted traffic classification based on multi-attribute Markov probability fingerprints. In *IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (pp. 1–10). IEEE. https://doi.org/10.1109/IWQoS.2018.8624182

[15] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (pp. 5998–6008). https://papers.nips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[16] Zejdl, P., Ubik, S., Macek, V., & Oslebo, A. (2008). Traffic classification for portable applications with hardware support. In *International Workshop on Intelligent Solutions in Embedded Systems* (pp. 1–9). IEEE. https://doi.org/10.1109/WISES.2008.4665073

[17] Park, J.-S., Yoon, S.-H., & Kim, M.-S. (2013). Performance improvement of payload signature-based traffic classification system using application traffic temporal locality. In *15th Asia–Pacific Network Operations and Management Symposium (APNOMS)* (pp. 1–6). IEEE. https://doi.org/10.1109/APNOMS.2013.6996521

[18] Sen, S., Spatscheck, O., & Wang, D. (2004). Accurate, scalable in-network identification of P2P traffic using application signatures. In *Proceedings of the 13th International World Wide Web Conference* (pp. 512–521). ACM. https://doi.org/10.1145/988672.988737

[19] Taylor, V. F., Spolaor, R., Conti, M., & Martinovic, I. (2018). Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, 13(1), 63–78. https://doi.org/10.1109/TIFS.2017.2736559

[20] Taylor, V. F., Spolaor, R., Conti, M., & Martinovic, I. (2016). AppScanner: Automatic fingerprinting of smartphone apps from encrypted network traffic. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroS&P)* (pp. 439–454). IEEE. https://doi.org/10.1109/EuroSP.2016.38

[21] Liu, C., He, L., Xiong, G., Cao, Z., & Li, Z. (2019). FS-Net: A flow sequence network for encrypted traffic classification. In *IEEE Conference on Computer Communications (INFOCOM)* (pp. 1171–1179). IEEE. https://doi.org/10.1109/INFOCOM.2019.8737553

[22] Cho, K., Van Merriënboer, B., Bahdanau, D., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation (arXiv:1406.1078). arXiv. https://arxiv.org/abs/1406.1078

[23] Korczynski, M., & Duda, A. (2014). Markov chain fingerprinting to classify encrypted traffic. In *IEEE Conference on Computer Communications (INFOCOM)* (pp. 781–789). IEEE. https://doi.org/10.1109/INFOCOM.2014.6847993

[24] Lashkari, A. H., Draper-Gil, G., Mamun, M. S. I., & Ghorbani, A. A. (2017). Characterization of Tor traffic using time-based features. In *Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP)* (pp. 253–262).

[25] Bhavya, P. S., Balaji, K., Banoth, N., & Parhamfar, M. (2024). A light weight Mobile Net SSD algorithm based identification and detection of multiple defects in ceramic insulators. *Journal of Modern Technology, 2024*, 59–74.

[26] Neelashetty, K., Goel, S., Inamdar, F., Dintakurthy, Y., Varanasi, L. N. S., & Krishna, V. B. M. (2025). Optimal energy management in

microgrids: A demand response approach with Monte Carlo scenario synthesis and K-means clustering. *International Journal of Computational and Experimental Science and Engineering*, *11*(1). https://doi.org/10.22399/ijcesen.1023

[27] Shbair, W. M., Cholez, T., Francois, J., & Chrisment, I. (2016, April). A multi-level framework to identify HTTPS services. In *Proceedings of the IEEE/IFIP Network Operations and Management Symposium* (pp. 240–248).

[28] Cheng, J., He, R., Yuepeng, E., Wu, Y., You, J., & Li, T. (2020, December). Real-time encrypted traffic classification via lightweight neural networks. In *Proceedings of the IEEE Global Communications Conference (GLOBECOM)* (pp. 1–6).

[29] Wang, W., Zhu, M., Wang, J., Zeng, X., & Yang, Z. (2017, July). End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In *Proceedings of the IEEE International Conference on Intelligence and Security Informatics (ISI)* (pp. 43–48).

[30] Lotfollahi, M., Siavoshani, M. J., Hossein Zade, R. S., & Saberian, M. (2020). Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, *24*(3), 1999–2012.

[31] Krishna, V. B. M., Melkeri, V. S., Goel, S., & Prasad, K. R. K. V. (2025). Two-stage energy management for maximizing renewable energy penetration. *Engineering Review*, *45*(1). https://doi.org/10.30765/er.2688

[32] Wang, W., et al. (2017). HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection. *IEEE Access*, *6*, 1792–1806.

[33] Rezaei, S., Kroencke, B., & Liu, X. (2020). Large-scale mobile app identification using deep learning. *IEEE Access*, *8*, 348–362.

[34] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778).

[35] Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies.

[36] Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

[37] Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *CoRR*, abs/1409.0473.

[38] Britz, D., Goldie, A., Luong, M. T., & Le, Q. V. (2017). Massive exploration of neural machine translation architectures. *CoRR*, abs/1703.03906.

[39] Draper-Gil, G., Lashkari, A. H., Mamun, M. S. I., & Ghorbani, A. A. (2016, February). Characterization of encrypted and VPN traffic using time-related features. In *Proceedings of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)* (pp. 407–414). Rome, Italy.

[40] Lin, X., Xiong, G., Gou, G., Li, Z., Shi, J., & Yu, J. (2022, April). 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). Lyon, France.

[41] Rogers, A., Kovaleva, O., & Rumshisky, A. (2020). A primer in BERTology: What we know about how BERT works. *Transactions of the Association for Computational Linguistics*, *8*, 842–866.

[42] Hubel, D. H., & Wiesel, T. N. (1968). Receptive fields and functional architecture of monkey striate cortex. *Journal of Physiology*, *195*, 215–243.

[43] Dos Santos, C., & Gatti, M. (2014, August). Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers* (pp. 69–78). Dublin, Ireland.

[44] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[45] Lotfollahi, M., Siavoshani, M. J., Hossein Zade, R. S., & Saberian, M. (2020). Deep packet: A novel approach for encrypted traffic classification using deep learning. *Soft Computing*, *24*, 1999–2012.

[46] Zhao, Z., Chen, H., Zhang, J., Zhao, X., Liu, T., Lu, W., ... & Du, X. (2019, November). UER: An open-source toolkit for pre-training models. In *Proceedings of EMNLP-IJCNLP 2019* (p. 241). Hong Kong, China.

[47] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[48] van Ede, T., Bortolameotti, R., Continella, A., Ren, J., Dubois, D. J., Lindorfer, M., ... & Peter, A. (2020, February). Flowprint: Semi-supervised mobile-app fingerprinting on encrypted network traffic. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, USA.

[49] Panchenko, A., Lanze, F., Pennekamp, J., Engel, T., Zinnen, A., Henze, M., & Wehrle, K. (2016, February). Website fingerprinting at internet scale. In *Proceedings of NDSS*. San Diego, CA, USA.

[50] Taylor, V. F., Spolaor, R., Conti, M., & Martinovic, I. (2017). Robust smartphone app identification via encrypted network traffic analysis. *IEEE Transactions on Information Forensics and Security*, *13*, 63–78.

[51] Al-Naami, K., Chandra, S., Mustafa, A., Khan, L., Lin, Z., Hamlen, K., & Thuraisingham, B. (2016, December). Adaptive encrypted traffic fingerprinting with bi-directional dependence. In *Proceedings of the 32nd Annual Conference on Computer Security Applications* (pp. 177–188). Los Angeles, CA, USA.

[52] Sirinam, P., Imani, M., Juarez, M., & Wright, M. (2018, October). Deep fingerprinting: Undermining website fingerprinting defenses with deep learning. In *Proceedings of the 2018 ACM SIGSAC*

Conference on Computer and Communications Security (pp. 1928–1943). Toronto, ON, Canada.

[53] Liu, C., He, L., Xiong, G., Cao, Z., & Li, Z. (2019, April). FS-Net: A flow sequence network for encrypted traffic classification. In *Proceedings of the IEEE INFOCOM 2019 - IEEE Conference on Computer Communications* (pp. 1171–1179). Paris, France.

[54] Shen, M., Zhang, J., Zhu, L., Xu, K., & Du, X. (2021). Accurate decentralized application identification via encrypted traffic analysis using graph neural networks. *IEEE Transactions on Information Forensics and Security*, *16*, 2367–2380.

[55] He, H. Y., Yang, Z. G., & Chen, X. N. (2020, December). PERT: Payload encoding representation from transformer for encrypted traffic classification. In *Proceedings of the 2020 ITU Kaleidoscope: Industry-Driven Digital Transformation (ITU K)* (pp. 1–8). Ha Noi, Vietnam.

[56] Shiravi, A., Shiravi, H., Tavallaee, M., & Ghorbani, A. A. (2012). Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, *31*(3), 357–374.

[57] Dintakurthy, Y., Innmuri, R. K., Vanteru, A., & Thotakuri, A. (2025). Emerging applications of artificial intelligence in edge computing: A comprehensive review. *Journal of Modern Technology*, *1*(2), 175–185.

[58] Al Barazanchi, I. I., Hashim, W., Thabit, R., & Hussein, N. A. K. (2024, March). Advanced hybrid mask convolutional neural network with backpropagation optimization for precise sensor node classification in wireless body area networks. *KHWARIZMIA*, *2024*, 17–31. https://doi.org/10.70470/KHWARIZMIA/2024/004

[59] Alazawi, S. A. H., Abdulbaqi, H. A., & Ali, A. H. (2024, August). CNN-based intrusion detection software for network operating system environment. *BJIoT*, *2024*, 79–86. https://doi.org/10.58496/BJIoT/2024/010

[60] Harinath, K. R., & Kumar, G. K. (2024). Encrypted network traffic classification and feature selection by ensemble of CNN and TLBO meta-heuristic algorithm. In N. Singh, A. K. Bashir, S. Kadry, & Y. C. Hu (Eds.), *Proceedings of the 1st International Conference on Intelligent Healthcare and Computational Neural Modelling. ICIHCNN 2022* (pp. xxx–xxx). Springer. https://doi.org/10.1007/978-981-99-2832-3_65