IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

IEEE SYSTEMS, MAN AND CYBERNETICS SOCIETY SECTION

Received 19 November 2023, accepted 10 December 2023, date of publication 14 December 2023,
date of current version 20 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3343189

RESEARCH ARTICLE

CBS: A Deep Learning Approach for Encrypted Traffic Classification With Mixed Spatio-Temporal and Statistical Features

MEHDI SEYDALI[1], FARSHAD KHUNJUSH[1], BEHZAD AKBARI[2], AND JAVAD DOGANI[1]

[1]Department of Computer Science and Engineering and Information Technology, School of Electrical and Computer Engineering, Shiraz University, Shiraz 71946-84334, Iran

[2]Department of Electrical and Computer Engineering, Tarbiat Modares University, Tehran 14115-194, Iran

Corresponding author: Farshad Khunjush (khunjush@shirazu.ac.ir)

ABSTRACT With the rapid advancement of the internet and online applications, traffic classification has become an increasingly significant topic in computer networks. Managing network resources, improving service quality, and enhancing cybersecurity are critical. Due to traffic encryption techniques, traditional traffic classification approaches have become ineffective and inaccurate. Therefore, the scientific community considers deep learning a high-performance approach for classifying encrypted traffic. This paper proposes an encrypted traffic classification approach, CBS, based on a deep learning technique. CBS can classify encrypted traffic at two levels using 1D-CNN, attention-based Bi-LSTM, and SAE deep network models. The proposed model classifies traffic types and applications based on a comprehensive set of session and packet-level features. CBS accurately distinguishes traffic classes using spatial, temporal, and statistical features extracted from packet content relationships, temporal relationships between packets in a session, and statistical characteristics of a work session. A traffic data augmentation technique based on a GAN network is employed to mitigate the impact of data imbalance on traffic classes. The proposed platform's performance is evaluated on the public ISCX VPN-Non VPN 2016 dataset. The results demonstrate that the platform accurately and efficiently identifies applications and classifies encrypted traffic. Compared to state-of-the-art methods, the proposed traffic classification model improves precision by 21.3%, accuracy by 13.1%, recall by 18.11%, and F1 score by 19.79%.

INDEX TERMS Deep learning, encrypted traffic, imbalanced data, packet features, traffic classification.

I. INTRODUCTION

Traffic classification is a technique that involves analyzing the data received from a packet or flow to identify different applications or types of traffic [1]. Traffic classification plays a critical role in network management tasks such as quality of service guarantee, optimal utilization of network resources, anomaly detection, malware traffic detection, and network intrusion detection. Accurate traffic classification tools are necessary for these tasks [2], [3]. Due to increasing privacy and security concerns for internet users, many applications use various types of traffic encryption [4]. Technology advancements necessitate the analysis of encrypted traffic for multiple reasons. Encrypted traffic conceals malicious activities, making detection and analysis more challenging. It also optimizes network performance by ensuring critical applications receive sufficient bandwidth and priority. Analyzing encrypted traffic is also crucial for regulatory compliance, as sensitive data in industries like healthcare and finance must be encrypted to comply with data privacy regulations [2], [3], [4]. Traffic classification approaches generally fall into four categories: port-based, payload-based, machine learning-based, and deep learning-based [5]. The first and most straightforward approach, port-based classification, identifies the type of traffic by extracting the port number from the packet header. Payload-based analysis,

The associate editor coordinating the review of this manuscript and approving it for publication was Huiyan Zhang.

known as Deep Packet Inspection (DPI), examines packet payloads against predefined patterns and signatures for various protocols [5], [6].

Both port-based and DPI approaches have certain limitations. The port-based approach could be more efficient at classifying traffic due to port obfuscation, random port number assignment, using dynamic ports, and the network address translation (NAT) technique [7], [8]. Furthermore, encryption techniques are widely employed in internet traffic to preserve privacy. Consequently, extracting useful information from the packet payload is less effective, which results in less accurate classification [9]. The primary issue with the DPI method is its high computing overhead, which makes it unsuitable for real-time or encrypted traffic classification [7], [10]. Therefore, statistical flow analysis and machine learning approaches have been investigated to overcome the limitations of the two previous methods.

Machine learning algorithms utilize time series and statistical information to learn distinguishing features and patterns in the traffic data [11], [12], [13]. Machine learning-based methods address issues with traditional approaches but introduce new challenges, such as capturing handcrafted packets and session features, which require domain specialists and can be time-consuming [14]. In other words, machine learning techniques are highly dependent on human-engineered features, which limits accuracy and generalizability [15], [16], [17]. A novel traffic classification method is required to address machine learning issues. Deep learning approaches have been proposed to handle machine learning challenges [8], [18], [19]. Unlike classical machine learning algorithms, deep learning algorithms perform feature extraction automatically, making them attractive for encrypted traffic classification.

Another advantage of deep learning methods is that they capture more complex patterns, resulting in more accurate classifications than machine learning methods [20], [21]. Deep learning has recently been widely applied in various fields, including speech recognition, computer vision (CV), image processing, and natural language processing (NLP) [22], [23], [24], [25]. Deep learning is a practical method for classifying encrypted traffic, often using Convolutional Neural Networks (CNNs) to extract features from traffic flow data. However, CNNs may not always be the best choice for extracting features from dynamic traffic flow data, which changes over time. These data often include statistics from work sessions, which CNNs often overlook. LSTM and Bi-LSTM models can extract features from traffic flow data. It is crucial to consider the temporal characteristics of the data and use a suitable technique for the traffic classification task. Despite the limitations of CNNs, they can still be used to extract features from traffic flow data [1], [12], [17], [18], [21]. Even though deep learning extracts features without human intervention, imbalanced learning data can result in poor results in this approach [26]. Semi-supervised sampling techniques are used to generate synthetic samples for datasets with limited sample sizes in specific classes to address data imbalance [27].

Feature engineering is a critical aspect of traffic classification, involving feature extraction and selection techniques [5], [7]. Selecting the most effective traffic features for the classifier is essential to achieving high classification accuracy, which depends on several factors [10]. The features directly influence the input structure and dimensions, which affect both computational complexity and the required number of packets for classification (memory complexity) [8], [10], [17]. Additionally, an appropriate model must be chosen based on the selected features. Various studies have classified encrypted traffic using spatial-temporal, statistical, or hybrid methods. However, due to the unique nature of encrypted traffic data, focusing on one feature type can lead to low classification accuracy. Deep learning networks offer precise traffic classification based on high-level features extracted from encrypted traffic flows or sessions. To understand the advantages of integrating temporal, spatial, and statistical features, they must first be defined as follows [9], [13], [22].

- **Temporal features:** Temporal features, such as packet frequency, size, and session duration, are time-dependent characteristics of encrypted traffic that help identify patterns and classify them accordingly.
- **Spatial features:** Spatial features are crucial in identifying the location of network packet bytes in encrypted traffic within a work session. These features capture the spatial relationships between bytes.
- **Statistical features:** Statistical features, such as average packet size, standard deviation, and inter-arrival time, are used to classify encrypted traffic and identify patterns, enabling efficient classification and analysis.

Combining temporal, spatial, and statistical features in encrypted traffic classification improves precision, efficiency, and robustness against various encrypted traffic types. This method provides deeper insight into traffic, enabling comprehensive analysis and identification of potential security threats [22]. Identifying encrypted traffic accurately is challenging due to its diversity. However, using multiple features can capture a broader range of traffic characteristics, making it more resilient to variations in encryption methods. This approach improves the interpretability of classification results and allows for a comprehensive understanding of traffic structure and behavior. It has flexible spatial, temporal, and statistical features [13], [14], [15], [22], [24]. Flexible features in encrypted traffic classification allow for classification regardless of the application or protocol, ensuring a comprehensive understanding of the nature of the traffic [10], [19].

This paper addresses the challenges of classifying encrypted traffic, which is often used by cybercriminals to conceal their activities. Data fragmentation and network congestion can also hinder precise classification. Each encryption algorithm and protocol have unique character-

istics, making identifying and classifying encrypted traffic challenging. As encryption algorithms and protocols evolve, new methods may emerge, making it crucial for traffic classification models to adapt. This research aims to improve classification accuracy by using multiple spatial, temporal, and statistical features and enhancing adaptability to new forms of encrypted traffic. The paper discusses the importance of using various features in encrypted traffic classification, especially in environments with incomplete or fragmented data.

Combining spatial, temporal, and statistical features can improve accuracy, especially when network congestion or other issues may result in fragmented traffic. By utilizing all three types of features, the proposed method can extract relevant information from partial or fragmented data, leading to more precise and reliable classification results. Adaptability is also crucial in determining and classifying new types of encrypted traffic. The model's ability to adapt is improved by using simultaneous spatial, temporal, and statistical features, enabling it to adjust to dynamic network conditions and respond to growing security threats. Classifying traffic based on traffic data's statistical, temporal, and spatial features is crucial for distinguishing traffic classes and application types.

This paper introduces a hybrid learning approach called CBS, which employs a comprehensive set of features to distinguish between different traffic classes effectively. CBS is a novel approach to classifying encrypted network traffic, leveraging an extensive array of features at both session and packet levels to support the classification of various traffic types and applications. The proposed method employs a traffic data augmentation technique based on a GAN network to generate synthetic samples for classes with limited samples, improving accuracy. CBS uses 1D-CNN, attention Bi-LSTM, SAE, and GAN as deep learning network models for feature extraction and data balancing. The extracted features are combined to represent the input data comprehensively. The proposed method uses 1D-CNN to learn spatial features from packet bytes. It uses attention-based Bi-LSTM to extract temporal sequence features between packets and SAE to compress extracted session statistics. Due to its comprehensive structure, CBS can classify encrypted and unencrypted traffic within a single platform.

CBS overcomes the limitations faced by existing methods, such as ineffective classification due to encryption techniques. By leveraging its unique feature selection approach, CBS provides an advanced solution for encrypted traffic classification. It automatically extracts features without handcrafted features, eliminating domain specialists' need to capture them manually. CBS adapts to new traffic patterns and encryption algorithms, making it a valuable tool for various applications, such as network security, traffic management, and quality of service monitoring. Our proposed method's main contributions can be summarized as follows:

- The CBS platform has been proposed to detect traffic at both levels: application identification and traffic characteristics.
- Traffic feature extraction and encrypted traffic classification have been combined into a single unified platform.
- The architecture of the proposed model includes three neural networks that extract local traffic features via 1D-CNN, temporal traffic features via Bi-LSTM, and traffic statistics features via SAE. Finally, these extracted features have been fused to form a comprehensive feature set.
- A GAN network has been employed to produce synthetic samples for classes with few instances. GAN network mitigates imbalanced data before feature extraction.
- The efficacy of the proposed model has been evaluated on a publicly available dataset. The results demonstrate that the proposed model performs well regarding precision, recall, F1 score, and accuracy.

The rest of the paper is organized as follows. Section II provides an overview of traffic classification research and related literature. Section III reviews the background of deep neural networks used in the proposed model. Section IV presents the proposed method for an encrypted traffic classification system. Section V describes the experimental setup. The results of traffic classification for traffic characterization and application identification are presented and analyzed in Section VI. Section VII discusses the results and advantages of the proposed CBS platform. Finally, the conclusion and future work are presented in Section VIII. The code and implementations are publicly available on GitHub.[1]

II. RELATED WORK

In recent years, the academic and industrial communities have paid much attention to traffic classification [12], [28], [29], [30], [31], [32]. Traditional techniques for traffic classification, like port-based and deep packet inspection (DPI), are no longer adequate due to the exponential growth in encrypted traffic [33], [34], [35]. Therefore, machine learning-based methods have been proposed to resolve the limitations of port-based and DPI approaches in identifying and classifying encrypted network traffic [36]. In recent years, deep learning for traffic classification has gained attention due to the limitations of traditional machine learning-based, such as manual feature extraction [8]. This section categorizes the most prevalent traffic classification methods into port-based, deep packet inspection, machine learning, and deep learning approaches.

A. PORT-BASED APPROACHES

In the past, port-based approaches were widely employed for traffic classification. This approach uses fixed and well-known port numbers assigned by the Internet Assigned Numbers Authority (IANA) for most communication protocols [37], [38], [39]. Further, this method extracts the TCP or UDP port number from the application traffic

[1] https://github.com/mehdiseydali/CBS.git

packet header. Subsequently, the port number is checked against the IANA-registered port numbers to classify the traffic [40]. Despite being the most straightforward approach for traffic classification, it has limitations, such as port obfuscation, dynamic ports, and network address translation (NAT). Therefore, this approach has low efficiency and accuracy.

B. DEEP PACKET INSPECTION (DPI) APPROACHES

The DPI approach, which is signature-based, analyzes the payload of the application layer. This approach aims to obtain predetermined patterns as signatures for every protocol and application. Subsequently, it classifies traffic by comparing the data in the packet's payload to the predefined signatures [6], [12], [41]. Although this method has high accuracy in classifying unencrypted traffic, it is computationally intensive. Moreover, this approach cannot identify and classify encrypted traffic and requires updating the signature library when a new protocol is developed [42], [43].

C. MACHINE LEARNING-BASED APPROACHES

This approach relies on statistical characteristics of traffic instead of packet payloads. These features include the minimum, maximum, and average packet sizes, the number of packets per flow, and the number of packet bytes per flow. This method analyzes these statistical features to gain insight into the nature and patterns of network communications without examining the specific data transmitted [44]. Using various feature extraction and selection methods to identify the optimal feature subset significantly affects the ability of machine learning algorithms to learn and classify traffic [45]. Numerous studies have been published on traffic classification using machine learning techniques. Dong et al. developed a Naïve Bayes-based algorithm to identify and categorize Skype network traffic, addressing insufficient labeling in real-time [46]. Additionally, an incremental support vector machine (ISVM) was utilized in [47] to reduce memory and CPU utilization during execution. This approach updates the classifier in response to newly acquired data. In [48], the authors utilized Principal Component Analysis (PCA) and the Genetic Algorithm (GA) to select significant features for the classification of SDN traffic. Gil et al. proposed the k-nearest neighbor (K-NN) and C4.5 decision tree algorithms to classify traffic based on time-related flow features [49]. An application-level traffic classifier was proposed in [50]. This study utilizes each application's unique payload size sequence signatures to identify its traffic. Zhai et al. used a random forest technique to classify SDN traffic. Using a random forest algorithm to generate random features leads to an optimal classification [51]. In [52], k-nearest neighbor (KNN) and k-means classification are combined to provide real-time encrypted traffic classification. This classification method relies on statistical features of the flow, and calculating such features in short-length flows is difficult. Therefore, flows containing less than 15 payload packets were excluded from the dataset. Five supervised machine learning algorithms that use statistical features to identify peer-to-peer traffic are proposed in [53]. This study applies supervised machine learning algorithms to classify flows based on statistical features extracted from packet streams, including packet headers [54]. classifies internet traffic flow using both supervised and unsupervised algorithms. This study employs an unsupervised learning approach to extract unidirectional and bidirectional flows from captured traffic, identifying statistically significant features and clustering flows based on their similarities. Finally, it utilizes these features as training data for a supervised learning engine to classify new and unforeseen traffic flows accurately. The main drawback of the machine learning approach is the need for expert assistance during the time-consuming and error-prone feature extraction and feature selection phases.

D. DEEP LEARNING-BASED APPROACHES

This approach is typically used for the classification of encrypted traffic. These networks can accept raw packets after preprocessing as input data. Although encrypted traffic contains hidden patterns, deep neural networks can extract significant features, enhancing the model's generalizability [10], [55]. Deep learning models, like convolutional neural networks (CNNs) and recurrent neural networks (RNNs), can effectively learn and extract complex patterns from network data. Automated network traffic categorization makes them useful for network management, cybersecurity, and quality of service planning [56]. In the following, we review deep learning applications in intrusion detection, malware detection, and encrypted traffic classification. In [57], an end-to-end platform based on one-dimensional convolution neural networks combines feature extraction and selection to classify encrypted traffic. This approach learns the spatial features of network traffic and proposes a session-based traffic classification. Cui et al. proposed a method for session-based traffic classification that utilizes a deep neural network to learn the spatial features of network traffic [58]. This study introduces a novel approach for classifying encrypted network traffic using CapsNet and session packets. The SPCaps technique is applied to reduce the weight of interference traffic and increase the weight of valuable traffic. CapsNet is then used to learn the spatial features of the encrypted traffic [58]. According to [59], Flowpic is a new method for classifying internet traffic that uses all time- and size-related information in a network flow instead of manually extracting features. Flowpic proposed a unique approach for identifying and classifying encrypted internet traffic and applications by transforming essential flow data into an intuitive image. The flow category is then determined using a CNN model based on the obtained images. Zeng et al. proposed TEST, a lightweight approach for traffic classification that extracts temporal and spatial features to classify encrypted traffic using a combination of CNN and LSTM networks. TEST employs a three-layer hierarchical framework to improve classification accuracy [60]. Wang et al. utilized 1D-CNN to classify malicious traffic.

This method presented an end-to-end system for classifying malicious traffic using a 1D-CNN network. It automatically extracts and selects features for classification purposes. The raw traffic is divided into session and flow levels and converted into a set of images before being classified by a 1D-CNN network [61]. Lotfollahi et al. proposed the classification of encrypted traffic using a combination of CNN and Stacked AutoEncoder (SAE). The proposed models in [8] aim to characterize traffic and identify applications. It generates a feature vector based on raw traffic and feeds it into SAE and 1D-CNN networks. The 1D-CNN model outperformed the SAE model based on the results [8]. In [62], a model called CLD-NET was introduced to detect unknown VPN network traffic flow accurately. This technique extracts features for classifying encrypted traffic by integrating packet payload and packet interval features. CNN and LSTM models were utilized to learn traffic statistical and sequence features. Bei et al. proposed a hybrid CNN-LSTM approach that extracts a session's temporal and spatial features using LSTM and CNN, respectively [63]. The features are merged and fed into a fully connected network for classification. ICLSTM resolves the data imbalance issue by assigning weights to various categories to enhance the data's symmetry [63]. In [64], a distinct feature called Sequential Message Characterization (SMC) is constructed from all flows associated with a specific application. Encrypted traffic is classified utilizing an LSTM and a multi-classifier. A Long Short-Term Memory (LSTM) neural network is established to learn features for classification. Message size sequences are fed into the LSTM, and deep LSTM models assign probability profiles to traffic types [64]. FlowGAN is a GAN-based method for classifying encrypted traffic that addresses the problem of data imbalance [65]. FlowGAN employs a GAN network to generate synthetic traffic samples for classes with a small number of samples and an MLP network to evaluate the effectiveness of traffic classification. FlowGAN utilizes a fully connected model to train the generator and discriminator of the GAN model. Pan et al. introduced the PacketCGAN approach, which uses a conditional GAN named CGAN to produce traffic data to address imbalanced classes [66]. PacketCGAN combines the synthetic data generated by the generator with the original data to create a new traffic dataset that maintains the balance between the major and minor classes in the dataset. A CNN-based semi-supervised method called DCGAN is introduced in [67]. This method employs a CNN network as the generator and discriminator to mitigate the impact of data imbalance on specific classes. In this study, the DCGAN model uses network packet time-series features, such as inter-arrival time, packet length, and packet direction, as input features. Table 1 provides an overview of the four reviewed approaches, while Table 2 summarizes the deep learning methods based on the proposed methodology, the dataset utilized, and the accuracy achieved.

The methods reviewed in this section have not comprehensively used all traffic features. Classifying encrypted traffic based only on spatial, temporal, or statistical features may not capture the complex patterns and characteristics of various types of traffic. Due to the lack of feature diversity, it is

TABLE 1. Overview of traffic classification.

Classification Approaches	Advantages	Disadvantages
Port-Based [12], [37]-[40]	• Simplicity • High Speed • Low Computational Cost	• Limitations such as port obfuscation, the usage of dynamic ports, and the use of network translating addresses (NAT)
DPI-Based [41]-[43]	• High accuracy in classifying unencrypted traffic	• High computational resource consumption • The requirement to update the signature library whenever a new protocol emerged • Inefficiencies in detecting and classifying encrypted traffic
Machine Learning-Based [44]-[54]	• Fast and consume less computational power	• Highly depend on an expert for the extraction and feature selection phases • Time-consuming and is prone to error due to human intervention
Deep Learning-Based [8], [57]-[67]	• Automatically performs feature extraction without human intervention • High learning ability in comparison with machine learning • DL is highly scalable	• Training Process is complex and slow • Requires Big Dataset

TABLE 2. Overview of deep learning traffic classification approaches.

Method	Methodology	Year	Dataset	Accuracy
[57]	1D-CNN	2017	ISCX VPN-Non VPN 2016	86.6% accuracy in 12 category classification
[58]	CapsNet	2019	ISCX VPN-Non VPN 2016	99.1% accuracy in 12 category classification
[59]	LeNet-5	2021	ISCX VPN-Non VPN 2016	average accuracy of 97.0% for non-VPN traffic, 99.7% for VPN
[60]	1D-CNN + LSTM	2019	ISCX VPN-Non VPN + USTC-TFC	99.98% accuracy on a dataset containing eight classes of traffic
[61]	2D-CNN	2017	USTC-TFC2016	99.41% average accuracy
[8]	1D-CNN , SAE	2017	ISCX VPN-Non VPN 2016	98% accuracy for application identification with 1D-CNN- For the traffic characterization CNN = 93% F1 and SAE = 92% F1
[62]	1D-CNN + LSTM	2021	ISCX VPN-Non VPN 2016	95% accuracy of 12 category classification
[63]	Inception CNN +LSTM	2021	ISCX VPN-Non VPN 2016	98.1% accuracy in 12 category classification
[64]	LSTM	2021	Private Dataset	98.9% accuracy in 8 category classification
[65]	MLP	2019	ISCX VPN-Non VPN 2016	99.1% accuracy for application identification
[66]	MLP	2020	SCX2012 + USTC-TFC2016	99.51% accuracy for application identification
[67]	2D-CNN	2019	ISCX VPN-Non VPN 2016 + QUIC	93% accuracy for both datasets

difficult to distinguish between similar types of traffic. Determining similar traffic types can be challenging due to limited feature diversity. The accuracy and effectiveness of encrypted traffic classification can be impeded by several limitations when classification is based only on spatial, temporal, or statistical features. By considering combined features and using various deep learning classification techniques, it is possible to improve the accuracy and effectiveness of encrypted traffic classification.

III. BACKGROUND ON DEEP NEURAL NETWORKS

Neural networks (NNs) are computational systems comprising numerous interconnected processing elements [68]. Typically, these networks consist of multiple building blocks known as neurons, which are connected through a series of links, each having its own weight [69]. Many data samples are utilized to train the neural network to produce the desired output during the learning process. Deep neural networks are a specific variant of neural networks with many hidden layers. Deep neural network learning platforms have gained popularity and attracted researchers' attention from diverse fields due to significant advances in computing power and the widespread availability of graphics processing units (GPUs) and tensor processing units (TPUs) [70], [71]. In the following sections, we will discuss the four deep neural network models, namely SAE, CNN, Bi-LSTM, and GAN, employed in our proposed method for encrypted traffic classification.

A. STACKED AUTOENCODER

AutoEncoders (AE) are unsupervised learning techniques that employ a feedforward neural network to reduce the dimensionality of input or extract features. This model aims to reconstruct the input data at the output with the least amount of reconstruction error possible between the input and output data [68], [72]. An autoencoder is composed of two parts: an encoder and a decoder. The encoder maps the input data to a lower-dimensional code, and the decoder transforms the lower-dimensional code back to the original data. Consider a learning dataset where xjϵRn. The objective of the autoencoder is defined as yj = xj for j = {1, 2, 3, ..., m}. Autoencoder tries to learn an objective function such that $Fw, b(x) \approx x$ where w represents the set of weights of the whole neural network, and b represents the biases vector. Equation (1) illustrates the generic form of the autoencoder error function, which aims to minimize the L error.

$$L(W, b) = min||x - Fw, b(x)||^2 \tag{1}$$

Three layers compose the autoencoder architecture: the input, hidden, and output layers. The hidden layer has considerably smaller dimensions than the input layer, and the output of the hidden layer, referred to as code, can be employed as a set of discriminative features for encrypted traffic classification tasks. The SAE architecture is proposed in order to improve performance and results [73]. This is accomplished by stacking many AEs into hidden layers, with one AE's output connected to the successive AE's input. Fig. 1 shows an AE with input, output, and a hidden layer.

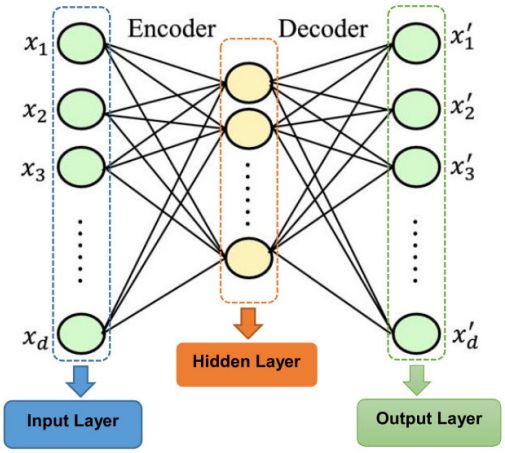


FIGURE 1. The autoencoder architecture.

B. CONVOLUTIONAL NEURAL NETWORK

Convolution neural networks (CNN) are essential deep neural networks that automatically extract input data features using the convolution operation [69], [72]. The convolution layer is the most significant component in a CNN network, as it receives an input of a square building block X of size N * N and a convolution kernel W of size m * m. The convolution operation produces an output layer Z of size (N-m + 1) * (N-m + 1). Equation 2 shows the function for computing Z, where f is a nonlinear activation function such as a rectified linear activation unit (ReLU) or a Leaky ReLU applied to the convolution output to learn more complex features of the input data.

$$Z_{ij} = f(\sum_{k=0}^{m-1} \sum_{l=0}^{m-1} W_{kl} X_{(i+k)(j+l)}) \tag{2}$$

The pooling layer, an essential block of CNN, is crucial in reducing output size and computational complexity, typically through max pooling or average pooling. CNNs have been used in various applications, including image processing, machine vision, and natural language processing [69]. Due to the sequential nature of network traffic, the 1D-CNN model was employed to classify encrypted network traffic. Wang et al. also demonstrated the superiority of the 1D-CNN model over the 2D-CNN model for classifying encrypted traffic in an experiment [57]. Using 1D-CNN, distinguishing features can be extracted for each traffic class and application based on spatial dependencies between adjacent bytes in network packets.

C. ATTENTION Bi-LSTM

As network traffic is a continuous stream of sequential data in packet bytes, deep neural networks, like LSTM and Bi-LSTM, have been proposed to extract temporal features between network packets [74]. As a particular type of RNN, LSTM networks address gradient vanishing and gradient

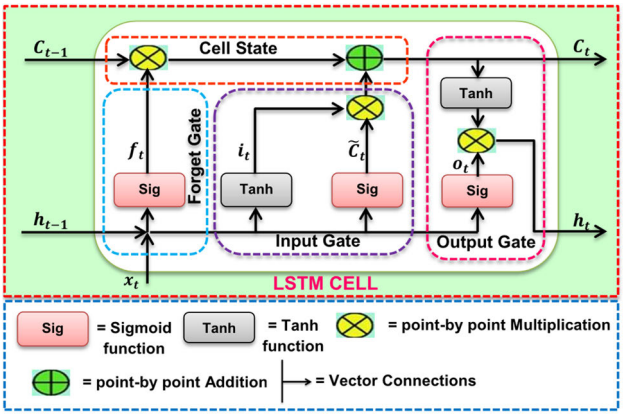


FIGURE 2. The LSTM cell structure.

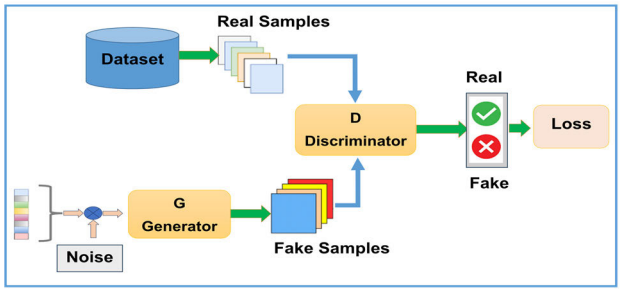


FIGURE 3. The GAN structure.

explosion as the two significant limitations of RNNs. Compared to RNNs, LSTM networks are much more efficient in identifying long-term dependencies within sequential data, making them the optimal option for extracting temporal features from network traffic data [72]. The LSTM structure consists of three gates: forget, input, and output, which are discussed below. Fig. 2 shows the fundamental components of an LSTM cell.

A forget gate controls the amount of information that should be discarded or retained. In a forget gate, data from the previous hidden state and the current input are passed through a sigmoid function, which generates a value between zero and one. As the value approaches zero, it is more likely to forget; as the value approaches 1, it is more likely to remember. Equation 3 calculates the forget vector, where W_f and b_f are the parameters of the forget gates, x_t is the input vector in step t, and h_{t-1} is the hidden state vector in step $t-1$.

$$f_t = \sigma\left(W_f . [h_{t-1}, x_t] + b_f\right) \tag{3}$$

The input gate acts as a value evaluator for data added to long-term memory as new information. This gate controls the amount of x_t input data added to C_t. In other words, it determines which data should be retained or forgotten for the cell state. The closer the input vector values are to one, the more data is kept in long-term memory; the closer the values are to zero, the more data is ignored. Equation 4 calculates the input vector, where W_i and b_i are the input gate parameters.

$$i_t = \sigma\left(W_i . [h_{t-1}, x_t] + b_i\right) \tag{4}$$

After calculating the input gate, the forgot gate vector f_t is multiplied by C_{t-1} (the previous step's cell state vector); then, the input gate vector is multiplied using $\tilde{C}_t$ pointwise multiplication, where $\tilde{C}_t$ represents the information contained in the hidden layer vector. Equation (5)-(6) gives the cell state C_t.

$$\tilde{C}_t = tanh\left(W_C . [h_{t-1}, x_t] + b_C\right) \tag{5}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{6}$$

The output gate in LSTM determines how much long-term memory should be transferred to the output. Equation (7)-(8) states how to calculate h_t, where W_o and b_o are the output gate parameters.

$$o_t = \sigma\left(W_o . [h_{t-1}, x_t] + b_o\right) \tag{7}$$

$$h_t = o_t * tanh\left(C_t\right) \tag{8}$$

Bi-LSTM networks are a variant of sequential processing models that consist of two LSTMs [75]. While one LSTM's input is in the forward direction, the other LSTM's is in the reverse direction. LSTM networks were developed to address long-term dependency. Because of its structure, LSTMs can memorize information over time and learn distant information exceptionally well. This study uses the Bi-LSTM model because of its higher prediction accuracy than the LSTM. The attention mechanism focuses on the information produced by the hidden layers of Bi-LSTM. Attention is a neural network mechanism that mimics cognitive attention by enhancing some input parts while disregarding others.

D. GAN ARCHITECTURE

In recent years, generative adversarial networks (GANs) have attracted attention as a form of deep learning. GANs consist of two primary neural network components: A Generator G(x) and a Discriminator D(x). The two components engage in an adversarial game against each other [26], [27]. The generator is a neural network that produces fake data to train the discriminator. The generator generates a sample using a random-length noise vector [76]. The generator's primary objective is to make realistic and sufficiently similar samples to the original. The discriminator is a neural network that distinguishes between real and fake data produced by the generator. The discriminator trains with data derived from two distinct sources. Real data samples are used as positive samples during the training process, while the generator's fake samples serve as negative samples. The generator, G, must transmit its generated output samples to the discriminator, D, for training. The discriminator distinguishes between authentic samples from the dataset and synthetic samples created by the generator. Simultaneously, the generator attempts to generate more realistic samples that could deceive D. The generator aims to minimize the discriminator's output while the discriminator strives to maximize it. Both components

FIGURE 4. The architecture diagram of CBS's proposed method.

engage in a competitive maximum-minimum game, which forms the fundamental basis for the adversarial nature of GAN. Fig. 3 depicts the structure of a GAN network and its two essential components.

IV. THE PROPOSED METHOD

This paper introduces a novel platform called CBS, as shown in Fig. 4. CBS consists of three components: traffic preprocessing, feature extraction, and classification. The traffic classification component uses deep learning models like 1D-CNN, Bi-LSTM, and SAE to classify traffic at application identification and characterization levels. The dataset's temporal, spatial, and statistical features are analyzed to identify packet dependency. Raw traffic is preprocessed based on session, which provides more information than flow-based separation, as shown in Fig. 5. A 1500-byte vector is produced after removing the MAC address, anonymizing the IP address, and normalizing the packet byte length. Twenty-five session-dependent statistical features are extracted from the traffic session. These 25 features are used as input to the SAE network.

Due to the original traffic format's hierarchical structure, 1D-CNN is chosen to extract the high-order spatial features of network sessions. Network packets are bytes transmitted through network channels, containing various forms of data such as headers, payloads, and control information. Through the dependency between adjacent bytes, a 1D-CNN can detect and extract meaningful patterns from a byte sequence: structures, characteristics, or behaviors relevant to a specific sequence. By analyzing these dependencies among contiguous bytes in packet data, 1D-CNNs can distinguish local relationships within packets. 1D-CNNs operate through convolution filters on input data, which are small weight matrices applied to a sliding input data window. Designing convolution filters with varying widths and depths allows capturing patterns of byte positions within packets. 1D-CNN can identify patterns at various levels of granularity, which are challenging to detect using conventional machine learning or rule-based methodologies. 1D-CNN can detect all packets with a TCP header with a specific sequence number, which is valuable for identifying all packets associated with a particular TCP connection. Further, 1D-CNN can recognize all packets associated with a specific application or service and all packets containing particular data types. For example, the following patterns could be learned from network packets by a 1D-CNN:

- TCP headers always begin with the same sequence of bytes.
- The payload always follows TCP headers.
- A TCP packet's sequence number is always more significant than the previous packet's sequence number.
- A TCP packet's acknowledgment number equals the next packet's sequence number.

Once 1D-CNN learns these patterns, it can recognize similar patterns in other sequences, as illustrated in Fig. 6. This information can classify packets or identify traffic flow patterns.

Bi-LSTM is a sequence processing model that analyzes network behavioral data and enhances the information available to the CBS model. It can extract temporal features from traffic sessions and predict application classes. Bi-LSTM is suitable for handling time series data, addressing dependency issues, and improving classification accuracy. Different packet classes, like chat and voice, have different inter-arrival time stamps. An attention-based mechanism assigns more weight to certain parts of the packet data while decreasing others. As illustrated in Fig. 6, Bi-LSTM is the primary model for learning the temporal characteristics of network traffic during traffic sessions. Bi-LSTM incorporates an attention mechanism to enhance long-term memory functions.

This study examines the unique characteristics of specific application classes based on session-level statistical features in network traffic. Network traffic is classified based on statistical features such as mean, standard deviation, minimum and maximum values, packet lengths, inter-arrival times, TCP flag counts, flow durations, and number of packets. Table 3 presents a list of statistical features employed in the SAE model. The proposed method assumes that each application's network layer traffic has unique statistical characteristics. In this paper, coding characteristics are used to extract features using SAE, which determines the application type but not the client type. Autoencoders are employed for dimensionality reduction and feature extraction. An autoencoder may not reduce the dimensionality of the input features due to complex relationships between the features in the dataset. Therefore, multiple autoencoders are employed to construct a stacked autoencoder. As illustrated in Fig. 6, the SAE network was employed to determine the relationships among the statistical features extracted from the session. Managing multiple developed applications presents challenges in network management, including acquiring sufficient training samples within a limited timeframe. Traffic samples suffer from class imbalances due to the varying popularity of different applications, leading to misclassification problems and performance decline. Deep learning-based algorithms can automatically extract traffic features but require massive data to learn traffic categories. To address this issue, CBS uses the GAN model to augment traffic data and generate targeted traffic, as shown in Fig. 5. Imbalanced datasets negatively impact deep learning models. Classes with more samples have higher accuracy, while minority classes have lower accuracy. The CBS platform deploys a GAN model before learning to mitigate the impact of data imbalance. This model

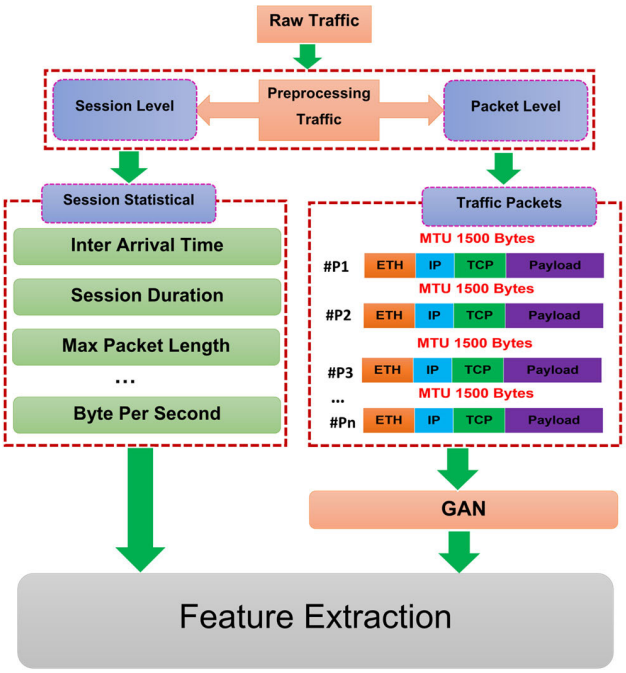


FIGURE 5. The architecture diagram of preprocessing step in the CBS model.

synthesizes samples like the original samples for classes with fewer samples (minority classes).

The 1D-CNN model is designed for extracting spatial features from packet bytes. Although it can improve traffic classification accuracy, it cannot extract statistical and temporal features among packets within a work session. In contrast, Bi-LSTM models can extract only temporal features from network traffic packets and cannot extract statistical or spatial features. Combining these three features is crucial for creating a comprehensive set of features for traffic classification. CBS's primary strength lies in integrating these features to accurately distinguish and classify network traffic packets. The traffic classification process combines extracted features to create a comprehensive set of features. As illustrated in Fig. 6 these features are fed into a fully connected network for traffic classification. Due to their structure-invariant nature, fully connected networks can classify traffic without requiring knowledge of the network structure. The softmax function converts a vector of N real numbers into a probability distribution of N possible classes. Figs 7 to 9 show the architectures of 1D-CNN, Attention Bi-LSTM, SAE, GAN, and FC models used in CBS. More details about datasets and traffic data preprocessing are provided in the following sections.

A. DATASET

This study evaluates the efficacy of the proposed methodology by utilizing the "ISCX VPN-Non VPN 2016" dataset. This dataset includes traffic from encrypted and unencrypted applications and provides two data levels for traffic classification. The first level involves identifying the type of application, such as Facebook or Skype. The second level involves identifying the protocol type, such as chat or email. Applications' traffic is divided into PCAP file formats and categorized by the application that generated it [49]. The ISCX VPN-Non VPN 2016 dataset comprises six classes of encrypted traffic and six classes of unencrypted traffic. Table 4 displays the various types of traffic and application-labeled content in the ISCX VPN-Non VPN 2016 dataset.

TABLE 3. List of the statistical features used in the SAE model of CBS.

No	Feature	Description
1	SBPS	Session bytes per second
2	SPPS	Session packet per second
3	Mean-pl	Mean packet length in a session
4	Min-pl	Min packet length in a session
5	Max-pl	Max packet length in a session
6	Std-pl	The standard deviation of packet length in a session
7	Mean-ps	Mean payload size in a session
8	Min-ps	Min payload size in a session
9	Max-ps	Max payload size in a session
10	Std-ps	The standard deviation of payload size in a session
11	Mean-IAT	Mean packet inter-arrival time in a session
12	Min-IAT	Min packet inter-arrival time in a session
13	Max-IAT	Max packet inter-arrival time in a session
14	Std-IAT	Standard deviation packet inter-arrival time in a session
15	Mean-AT	Mean active time of a session
16	Min-AT	Min active time of a session
17	Max-AT	Max active time of a session
18	Std-AT	The standard deviation of the active time of a session
19	Mean-IT	Mean idle time of a session
20	Min-IT	Min idle time of a session
21	Max-IT	Max idle time of a session
22	Std-IT	The standard deviation of idle time of a session
23	TP	The total number of packets truncated in a session
24	SPN	The total number of packets in a session
25	DS	Duration of the session

B. DATA PREPROCESSING

Data preprocessing is crucial for a deep learning model to interpret input packets effectively, as shown in Fig. 10. Preprocessing raw data traffic at both packet and session levels provides a comprehensive understanding of network traffic. Packet-level preprocessing is essential for feature extraction from individual packets, providing granular insights into specific patterns. Preprocessing at the session level is crucial for extracting session-specific features because encrypted traffic conceals its content at the packet level, making packet-level analysis challenging. Both preprocessing approaches enable a better understanding of network traffic, allowing more accurate traffic classification and valuable insights into network behavior. Preprocessing data at both levels enables the extraction of a broader spectrum of features and the detection of more complex traffic patterns. This contributes to more precise and reliable traffic classification results. As outlined

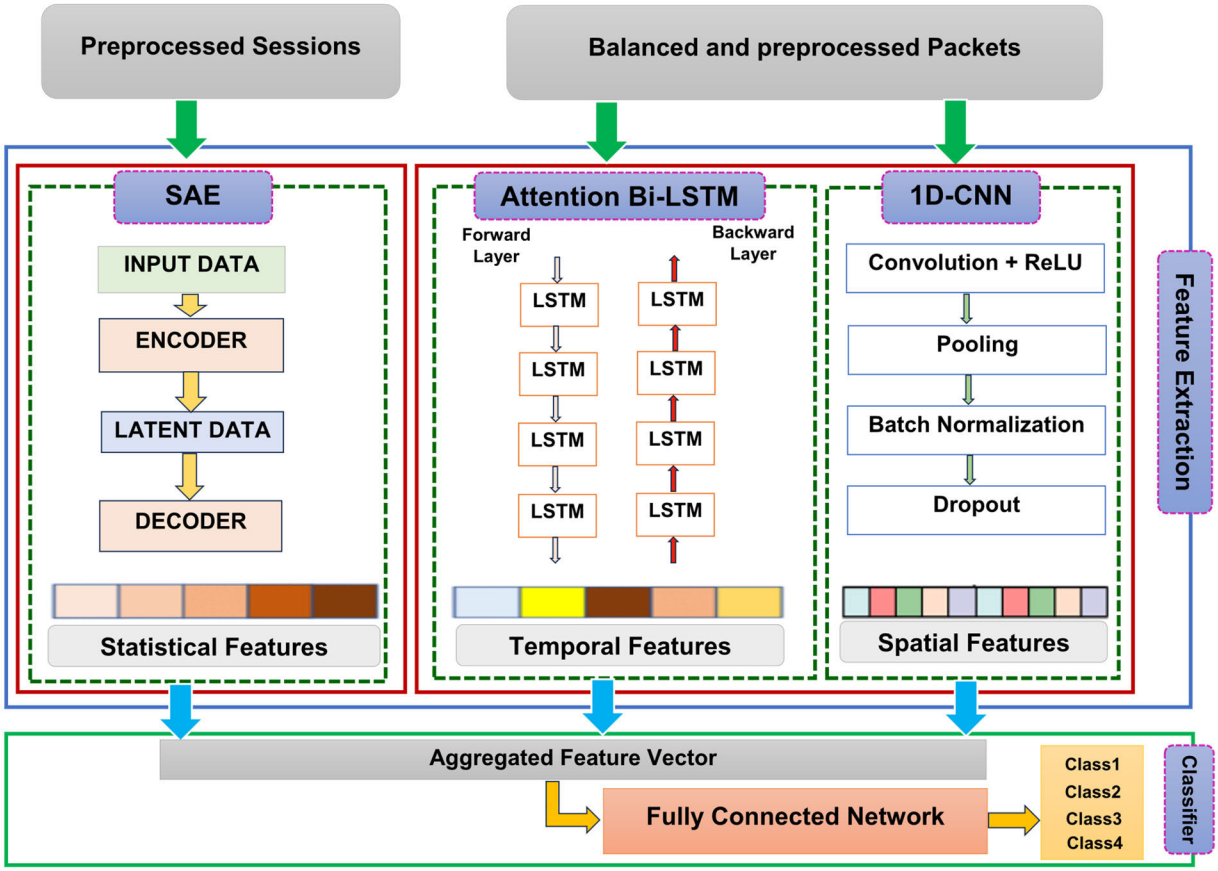


FIGURE 6. The architecture diagram of the feature extraction and classification step in the CBS model.

TABLE 4. Classes of ISCX VPN-Non VPN 2016 dataset.

Class Encryption Type	Traffic Type	Applications	Total of Sessions	Total of Packets
Non-VPN Traffic	Chat	Icq/AIM/Facebook/Hangout/Gmail/Skype	10806	11K
	Email	Email/Gmail	7761	12K
	File Transfer	FTPS/SFTP/SCP/Skype	69225	19K
	P2P	Torrent/Tor	1045	13K
	Streaming	YouTube/Netflix/Spotify/Vimeo/Skype	2260	21K
	VoIP	Skype/Voipbuster/Hangout/Facebook	199302	21K
VPN Traffic	VPN-Chat	Icq/AIM/Facebook/Hangout/Gmail/Skype	4029	13.7K
	VPN-Email	Email/Gmail	298	6.8K
	VPN-File Transfer	FTPS/SFTP/SCP/Skype	1020	17.5K
	VPN-P2P	Torrent/Tor	477	6K
	VPN-Streaming	YouTube/Netflix/Spotify/Vimeo/Skype	659	12K
	VPN-VoIP	Skype/Voipbuster/Hangout/Facebook	12285	14K

below, packet and session-level preprocessing involve five and three steps. The steps performed in the packet-level preprocessing phase are as follows:

(1) Non-Usable Packet Removal: Real-world datasets may contain packets that do not provide helpful information for deep learning. At the packet level, packets lacking valuable information are eliminated to ensure effective traffic classification. For instance, TCP connection segments with SYN, ACK, or FIN flag sets are excluded, as they don't provide valuable application information. Moreover, TLS and DNS key exchange segments that are not useful for identifying applications or classifying traffic should be removed.

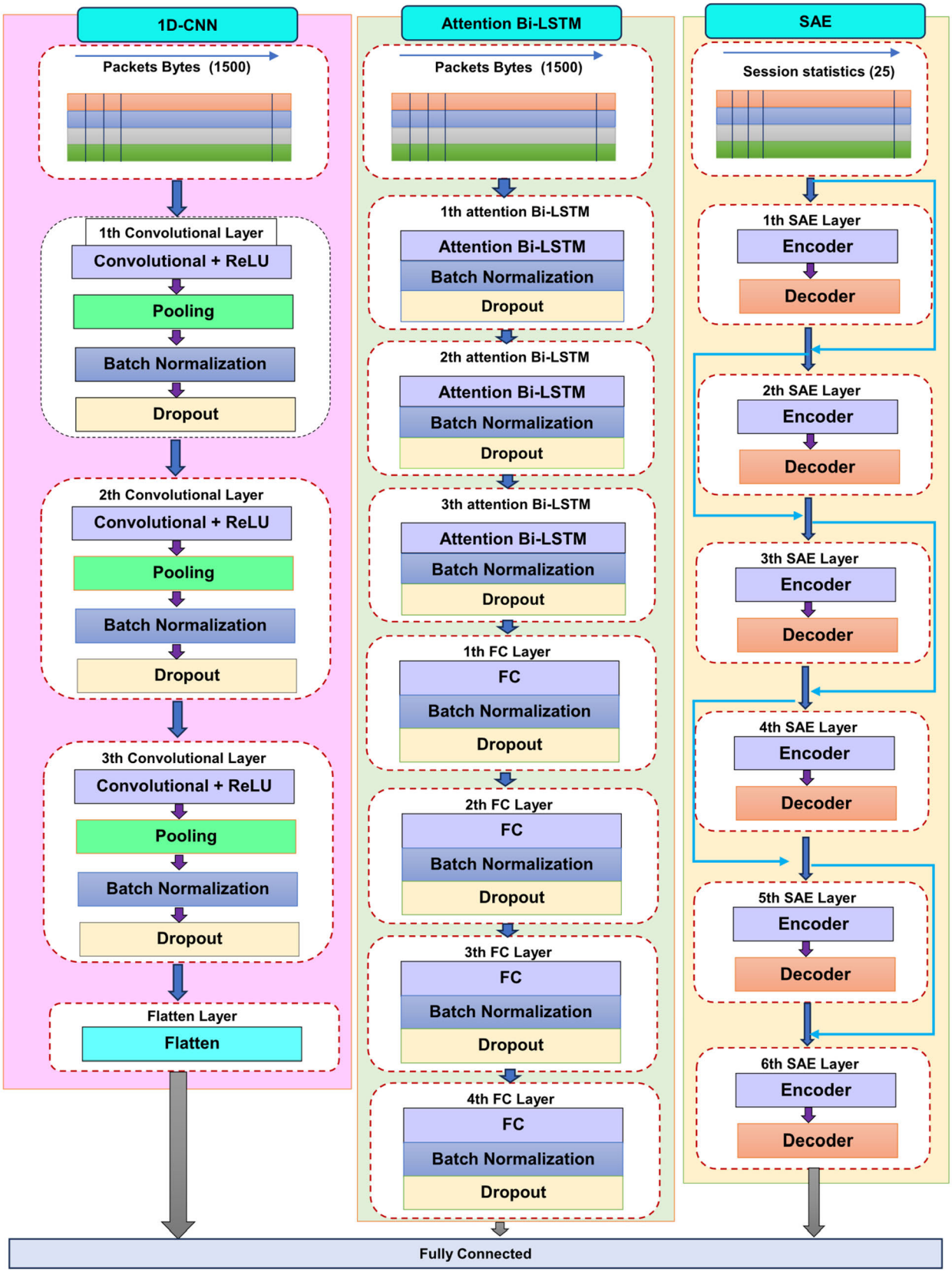


FIGURE 7. The architecture diagram of the 1D-CNN, the attention Bi-LSTM, and the SAE models for feature extraction in the CBS model.

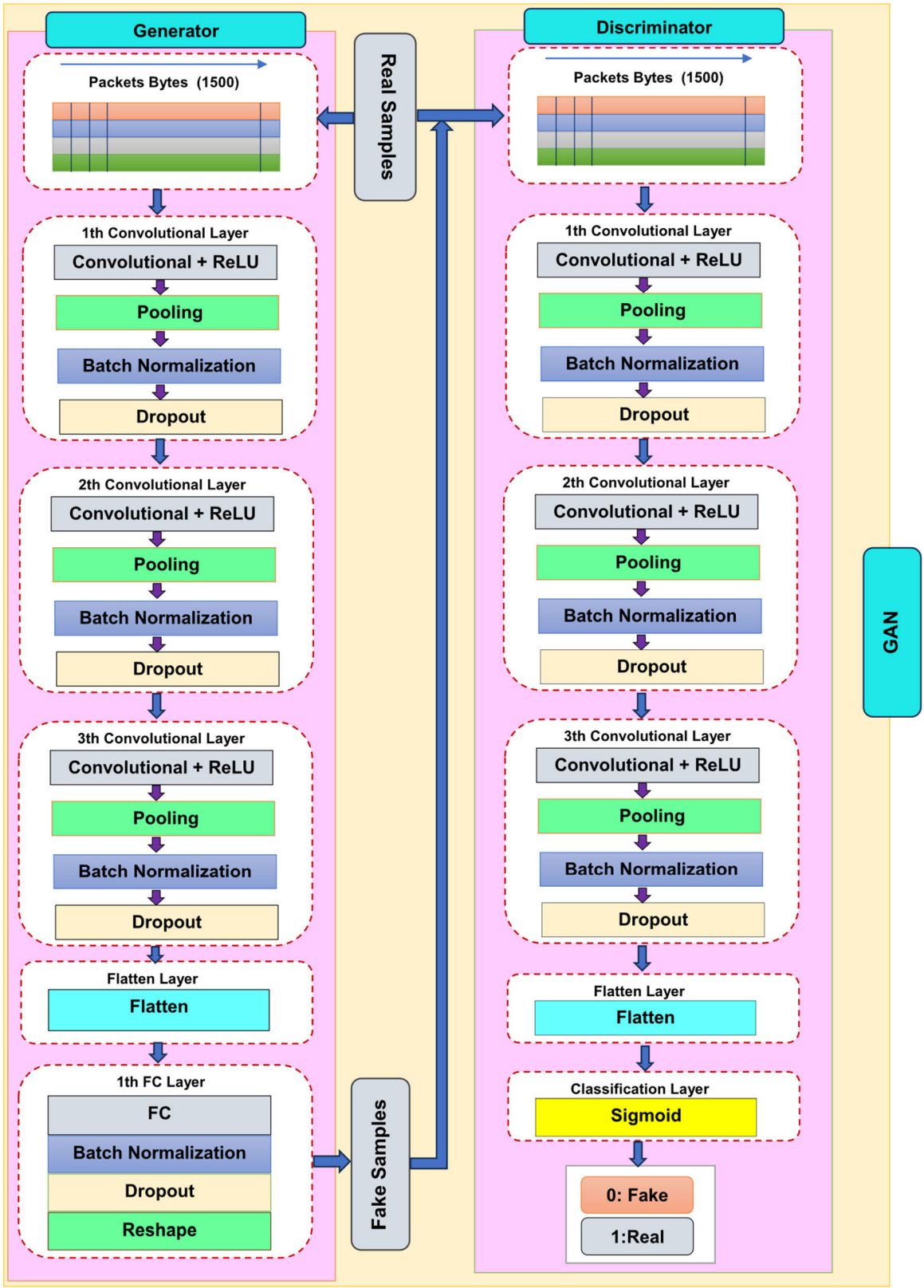


FIGURE 8. The architecture diagram of the GAN model for data imbalance in the CBS model.

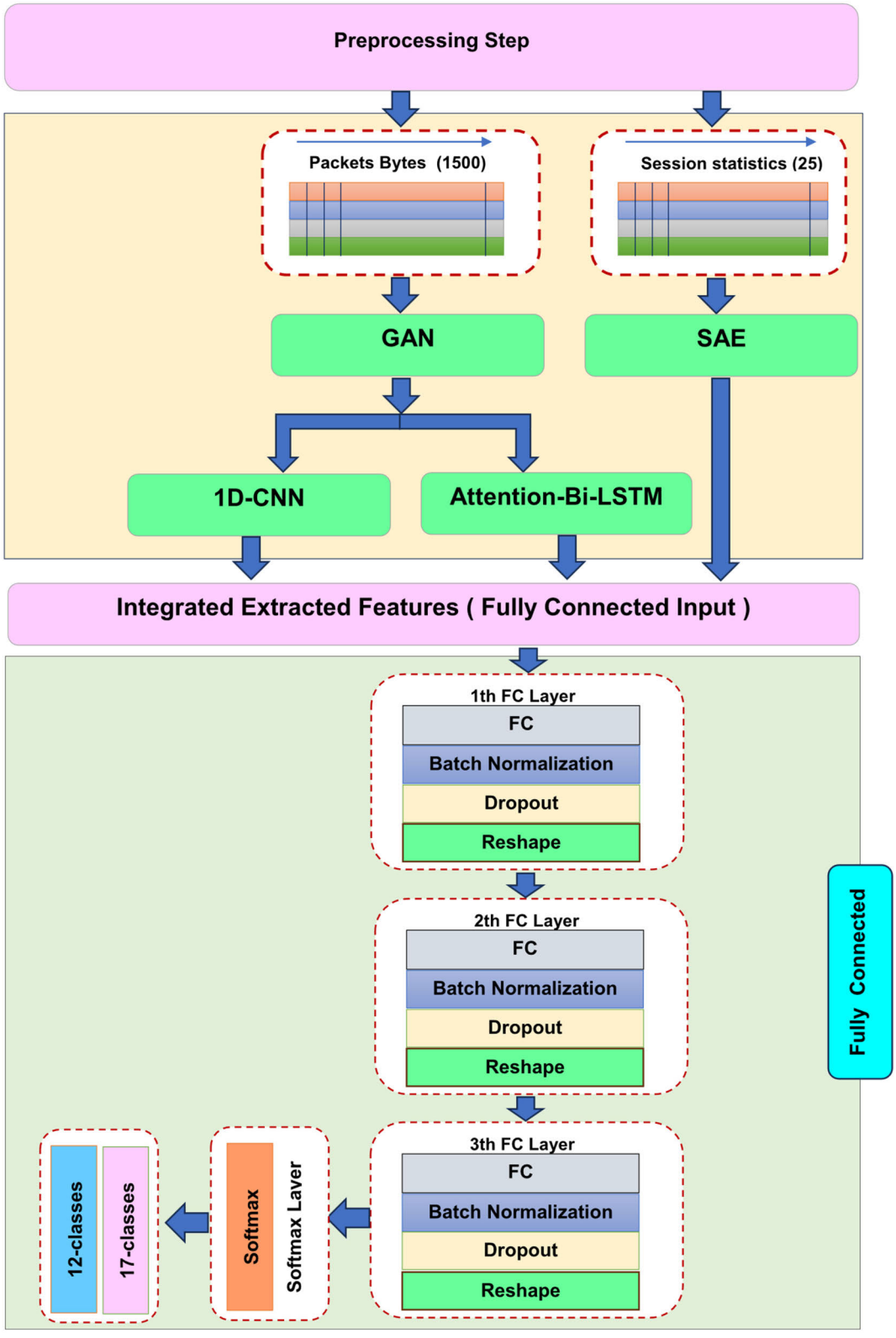


FIGURE 9. The architecture diagram of the FC model for traffic classification in the CBS model.

(2) Data Link Header Removal: Data link layer headers, which include MAC addresses, do not provide any relevant information for traffic classification. Therefore, this information is eliminated from the packet.

(3) IP Header Anonymization: In a DNN network, packets may be classified based on IP addresses, so packet IP headers must be anonymized through masking to prevent overfitting. Anonymization ensures that the DNN network does not classify traffic based on irrelevant features.

(4) Normalization and Unifying: In data normalization, each byte is converted into an individual number and fed into a Deep Neural Network (DNN). Unification converts data into a common representation for encrypted traffic, facilitating the design and implementation of deep learning algorithms for traffic classification. Data scaling helps to reduce overfitting and improves generalization performance, making algorithms more effective on unseen data. Data normalization can be achieved by transforming data into a range of values, such as [0-1]. Scaling packets from [0-255] to [0-1] will ensure equal impact on the learning model, optimizing performance. This prevents one byte from disproportionately affecting the model. For example, a packet containing bytes [214, 76, 87, 95] would assign a higher weight to the first byte (214) due to its greater value. When scaled to [0-1], the packet would have [0.839, 0.298, 0.341, 0.372], resulting in all three bytes affecting the learning model equally, ensuring optimal performance.

(5) Truncating – Zero Padding: A DNN cannot calculate gradient descent efficiently if the input data does not follow a fixed size, making it difficult or impossible to compute. This ensures consistency in the network architecture and batch data processing. However, network traffic and packet-level data pose significant challenges when dealing with variable data lengths, making input of such data directly into the DNN impractical. Data truncation and zero-padding are techniques used to achieve fixed-size inputs for DNN networks. Data truncation reduces packet lengths to a fixed size, while zero-padding is used when the original length falls short of the desired size. Zero padding ensures uniform inputs for packets of varying lengths, facilitating compatibility with the DNN architecture.

In Fig. 11, the raw input data to the figure's left exceeds the desired input size, so truncation is required. In contrast, the raw input data on the right side of Fig. 11 must exceed the expected input size, necessitating zero padding. Packet length statistics were analyzed to determine the ideal fixed truncation length. Studies indicate that most packets carry a payload of no more than 1500 bytes, also known as Maximum Transmission Units (MTU) [8]. As an example of packet length analysis, the Skype audio1a.pcap file was used. As shown in Fig. 12, the probability mass function is displayed for packet lengths ranging from 0 to 1500 bytes. The packet length distribution function was derived for all PCAP files, and the aggregate length distribution function is depicted in Fig. 13. DNN networks are fed 1500 bytes of normalized data as input vectors following steps 1-4. Preprocessing traffic packets at the session level involves the following steps:

(1) Split Session: A flow is a sequence of packets with the same source, destination, port, and protocol, whereas a session allows IP addresses to be exchanged between both directions. PCAP files are divided into sessions since sessions provide a better classification of traffic and statistical features than flows.

(2) Extract Session Features: This step extracts statistical features from raw traffic to differentiate traffic types and applications. Statistical features like packet count, session duration, and maximum payload size are included in these features. Table 3 provides a summary of the statistical features used in this paper.

3) Convert to Vector: Using the Min-Max method, the collected data falls within the range of [0, 1] based on a session's statistical features. The output of a SAE network is converted to feature vectors.

Fig. 14 shows an example of packet preprocessing. As shown in Fig. 14, a packet is converted to a vector of length 1*1500 after preprocessing steps. Preprocessed packets are vectors of numbers, where each number represents a packet feature. Preprocessed packets are then fed into a deep-learning model. Raw session data is preprocessed to extract relevant features before feeding the preprocessed features to the deep learning model for classification. Fig. 15 shows an example of session preprocessing. In Fig. 15, a single session is converted to vectors of length 1*25 after preprocessing.

V. EXPERIMENT

This section demonstrates the experimental setup, evaluation metrics, experiment types, proposed model parameters, and hyperparameters used in the proposed method.

A. EXPERIMENTAL SETUP

This paper implements the proposed model using the Keras library, with TensorFlow as its backend engine. The specific software and hardware employed for implementing the model are summarized in Table 5 as environment parameters.

B. EVALUATION METRICS

This paper employed four performance metrics to evaluate the experimental results: accuracy (Acc), precision (Pr), recall (Re), and F1 score (F1). The definitions of these metrics are provided below. (1) True positives (TP) refer to the number of instances that belong to a class and are predicted to belong to the same class by the model; (2) False positives (FP) denote the number of instances that do not belong to a class but are predicted to belong to the same class by the model; (3) True negatives (TN) indicate the number of instances that do not belong to a class and are predicted not to belong to that class by the model, and (4) False negatives (FN) represent the number of instances that do belong to a class but are misclassified by the model to belong to other

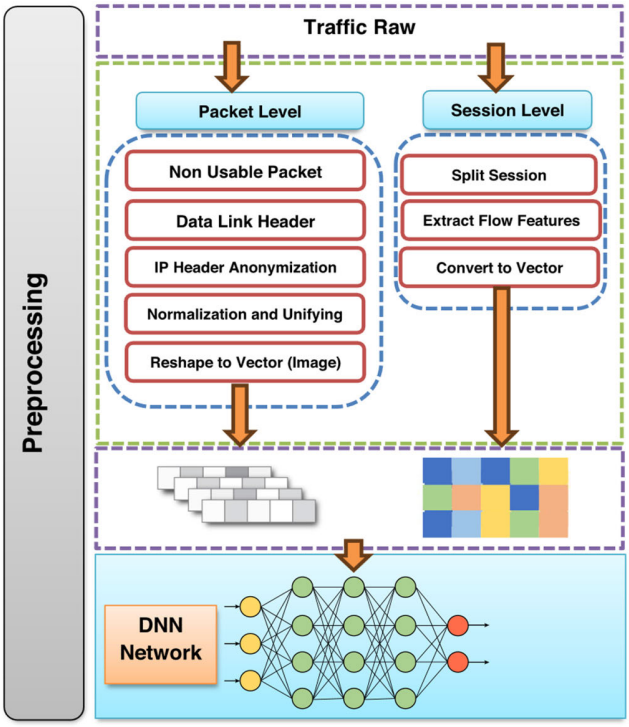


FIGURE 10. The network traffic preprocessing diagram.

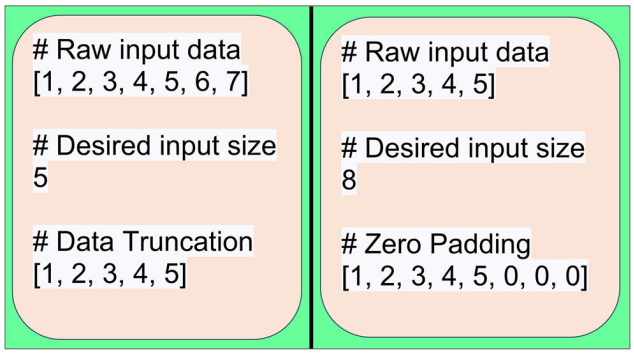


FIGURE 11. An example of truncation and zero padding.

classes. Consequently, the four aforementioned evaluation metrics can be defined as follows.

- Accuracy (Acc) $= \frac{TP+TN}{TP+TN+FP+FN}$
- Recall (Re) $= \frac{TP}{TP+FN}$
- Precision (Pr) $= \frac{TP}{TP+FP}$
- F1 Score $= \frac{2*Pr*Re}{Pr+Re}$

C. EXPERIMENT TYPE CONFIGURATION

The performance of the proposed model was evaluated through four experimental scenarios. The first experiment aimed to classify encapsulation protocols commonly employed to distinguish between VPN and Non-VPN traffic. The second and third experiments focus on traffic classification for six classes in both encrypted and unencrypted modes, respectively. The fourth experiment addressed a

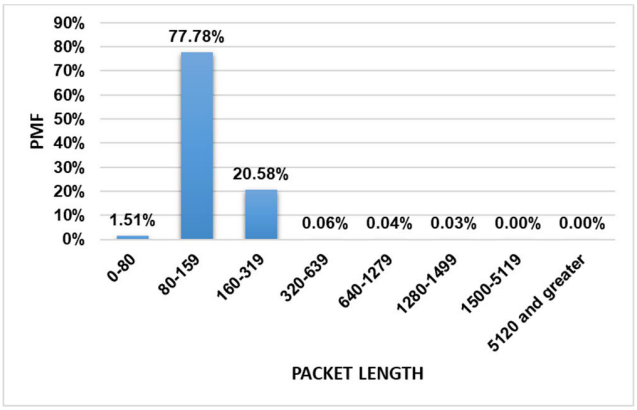


FIGURE 12. The packet length distribution of skype_audio1a.pcap.

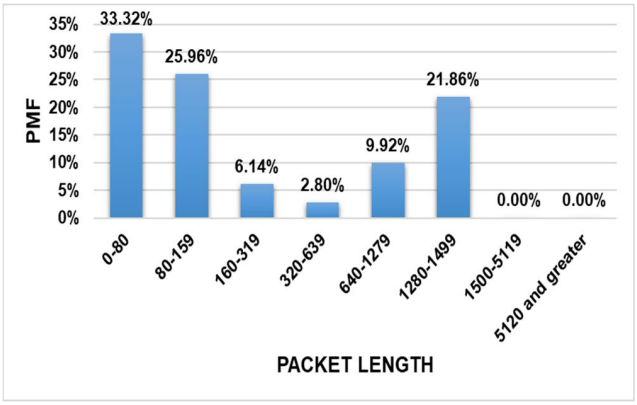


FIGURE 13. The packet length distribution of all files of the dataset.

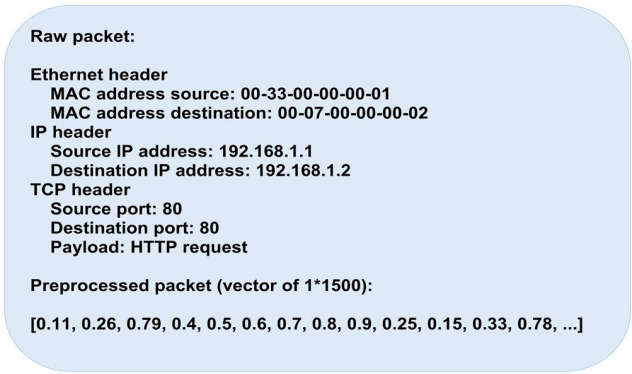


FIGURE 14. An example of packet preprocessing.

twelve-category classification problem that classified regular and encrypted traffic. Table 6 summarizes the experiments mentioned above.

D. EXPERIMENT HYPERPARAMETERS OF THE PROPOSED METHOD

In deep learning, a hyperparameter is a parameter set before the learning process and is distinct from other model parameters specified during the learning process. Our model's

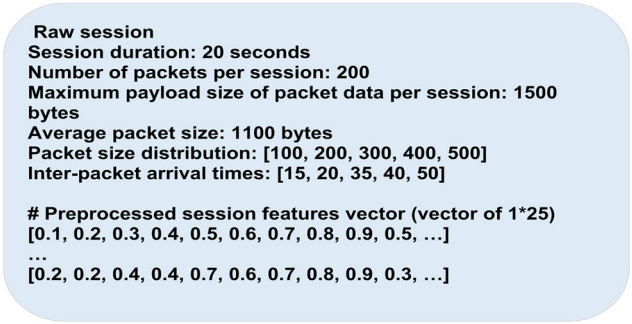


FIGURE 15. An example of session preprocessing.

TABLE 5. The experiment setup.

Configuration List	Specification
Operation System	Ubuntu 20.04 Lts 64-bit
CPU	Intel® Xeon® X5560 @ 2.80 GHz (4 processors)
GPU	GeForce® 1050 Ti
RAM	16G
Deep Learning Library	Keras
Deep Learning Backend	TensorFlow
Cuda Tools	NVIDIA Cuda9
cuDNN Version	NVIDIA cuDNN9
Packet Analysis Library	Scapy
Python Version	3.8

hyperparameters include the learning rate, loss function, and optimizer function. We utilized the dropout operator to prevent overfitting in deep neural network models. The dropout probability value during the learning process was set to 0.4. The initial learning rate value is a crucial factor that affects the model's performance. A low value results in a long training time and overfitting, while a high value leads to a fast training time but underfitting. For all our experiments, we set the initial value of this parameter to 0.001. The batch size, epoch, optimizer, and loss function were respectively set to 256, 50, Adam, categorical cross-entropy. The batch normalization technique employed in CBS involves normalizing the data so that the data points are adjusted to have a mean value of zero and a standard deviation of one. The ReLU activation function was applied in all CBS model layers. Table 7 summarizes the hyperparameters used in this study. We divided the dataset into two parts for each experiment, with 80% of the data used for training and 20% for evaluating the proposed model. We employed the ten-fold cross-validation method for validation, with nine parts for learning and one for testing.

E. MAIN PARAMETERS OF THE PROPOSED MODEL

Our proposed method employs five deep neural network models: the 1D-CNN network, the attention-based Bi-LSTM network, the SAE network, the FC, and the GAN. The structures of these networks and their primary parameters are

TABLE 6. The description of experiments scenario.

Experiment	Description	Classifier
1	Protocol Encapsulated Traffic Identification	2-Classes
2	Regular Traffic classification (Non-VPN)	6-Classes
3	Protocol Encapsulated Traffic Classification (VPN)	6-Classes
4	Encrypted Traffic Classification (VPN and Non-VPN)	12-Classes

TABLE 7. The values of hyperparameters in the CBS model.

Parameter Name	Dropout	Learning Rate	Batch Size	Epoch
Value	0.4	0.001	256	50
Parameter Name	**Optimizer**	**Loss Function**	**Activation Function**	**Batch Normalization**
Value	Adam	Categorical Cross Entropy	ReLU	[77]

presented in Table 8. The sequential nature of data packets in network traffic flow necessitates using a 1D-CNN network, which outperforms a 2D-CNN network [57], [78]. The 1D-CNN and Bi-LSTM networks have 1500 input features, while the SAE network has only 25. 1D-CNN is used in the construction of the generator and discriminator of GAN. The SAE network's latent layer yields ten extracted statistical features combined with 1D-CNN and Bi-LSTM network features. Finally, 1300 features are extracted from the three models and fed into a fully connected network to classify the encrypted traffic type and identify the application. There are 12 traffic types and 17 applications in total.

VI. EXPERIMENT RESULTS ANALYSIS

This section presents the results of several experiments conducted to assess the effectiveness of the proposed model. The system's performance is compared to other methods using the evaluation metrics outlined in section V.

A. THE CBS VALIDATION

We individually assessed each model's efficacy for traffic classification to highlight the advantages of fusing the models depicted in Fig. 5 and Fig. 6. Nevertheless, the experiments to evaluate the models' effectiveness showed that they cannot accurately classify encrypted traffic individually. Table 9 presents the results of these experiments conducted on the dataset evaluated by each model. The input data length required for attention-based Bi-LSTM and 1D-CNN models is 1500. In contrast, the SAE model feeds statistical features extracted from traffic files and sessions into the

TABLE 8. The main parameters of the CBS model.

Model	Layer Name	Operation	Input	Filter	No of Filter	Stride	Output
CNN	Conv1-Relu	Convolution+Relu	1*1500	1*7	256	1	1494
	Pooling1	AveragePooling	1494	1*6	-	6	249
	Conv2-Relu	Convolution+Relu	249	1*6	128	1	244
	Pooling2	AveragePooling	244	1*4	-	4	61
	Conv3-Relu	Convolution+Relu	61	1*5	64	1	57
	Pooling3	AveragePooling	57	1*3	-	3	19
	Flatten1	flatten	19	-	-	-	1216
Attention Bi-LSTM	A-Bi-LSTM1	Attention-Bi-LSTM	1500	-	-	-	1500
	A-Bi-LSTM2	Attention-Bi-LSTM	1500	-	-	-	1500
	A-Bi-LSTM3	Attention-Bi-LSTM	1500	-	-	-	1500
	FC1	Fully Connected	1500	-	-	-	1024
	FC2	Fully Connected	1024	-	-	-	512
	FC3	Fully Connected	512	-	-	-	256
	FC4	Fully Connected	256	-	-	-	74
SAE	SAE1	Fully Connected	25	-	-	-	20
	SAE2	Fully Connected	20	-	-	-	15
	SAE3	Fully Connected	15	-	-	-	10
	SAE4	Fully Connected	10	-	-	-	15
	SAE5	Fully Connected	15	-	-	-	20
	SAE6	Fully Connected	20	-	-	-	25
GAN (Generator)	Conv1-Relu	Convolution+Relu	1*1500	1*3	64	1	1498
	Pooling1	AveragePooling	1498	1*2	64	2	749
	Conv2-Relu	Convolution+Relu	749	1*3	32	1	747
	Pooling2	AveragePooling	747	1*2	32	2	373
	Conv3-Relu	Convolution+Relu	373	1*3	16	1	371
	Pooling3	AveragePooling	371	1*2	16	2	185
	Flatten1	flatten	185	-	-	-	2960
	FC1	Fully Connected	2960	-	-	-	1500
GAN (Discriminator)	Conv1-Relu	Convolution+Relu	1*1500	1*3	64	1	1498
	Pooling1	AveragePooling	1498	1*2	64	2	749
	Conv2-Relu	Convolution+Relu	749	1*3	32	1	747
	Pooling2	AveragePooling	747	1*2	32	2	373
	Conv3-Relu	Convolution+Relu	373	1*3	16	1	371
	Pooling3	AveragePooling	371	1*2	16	2	185
	Flatten1	flatten	185	-	-	-	2960
	FC1	Fully Connected	2960	-	-	-	1
FC	FC1	Fully Connected	1300	-	-	-	1000
	FC2	Fully Connected	1000	-	-	-	512
	FC3	Fully Connected	512	-	-	-	256
	Softmax1	Softmax	256	-	-	-	12-(17)

encoder-decoder network. Each model extracts the features and employs them as input for a DNN network and a softmax function to classify traffic. Table 9 summarizes the results of Experiments 2 and 3 conducted on the models included in the proposed architecture. However, these models could not accurately classify traffic based on the evaluation metrics described in Section V.

Separated models can only extract specific temporal, spatial, and statistical features. Utilizing only a subset of these features may result in inadequate traffic classification. However, the efficiency of the proposed approach is demonstrated by evaluating how integrating these models impacts the results. Table 10 illustrates the effect of integrating the models employed in the proposed approach. As depicted in Table 10, combining these models improves efficiency by simultaneously utilizing temporal, spatial, and statistical features. Combining features has led to more effective traffic classification. Table 10 shows Experiment 4 results for integrating spatial, temporal, and statistical features, which merge VPN and Non-VPN modes.

B. APPLICATION IDENTIFICATION AND TRAFFIC CHARACTERIZATION RESULTS

Table 11 shows CBS' performance results. The results show that each experiment's accuracy, precision, recall, and F1 score are higher than 99.21%. The results indicate that our model can extract essential and practical features for traffic and application classification and demonstrate the effectiveness of the proposed model. As shown in Table 11, the two-category classes have accuracy, precision, and recall higher than 99.67%, with an F1 score higher than the six- and twelve-category classes. The F1 score for classifying

TABLE 9. The performance of the separated models in the CBS platform.

Type	Attention Bi-LSTM				1D-CNN				SAE			
	Acc	Pr	Re	F1	Acc	Pr	Re	F1	Acc	Pr	Re	F1
Chat	77.2	75.13	72.7	73.90	73.33	72.75	70.25	71.48	68.21	70.13	69.2	69.66
Email	83.14	80.11	78.83	79.46	81.65	77.34	74.46	75.87	75.2	74.12	73.52	73.82
File Transfer	88.35	87.13	90.15	88.61	89.11	90.55	91.15	90.85	80.3	79.73	77.7	78.70
P2P	86.78	89.45	87.37	88.40	88.28	87.64	89.27	88.45	82.5	80.45	79.15	79.79
Streaming	85.17	86.34	83.89	85.10	83.65	84.34	86.31	85.31	81.7	82.67	81.76	82.21
VoIP	81.7	84.03	83.14	83.58	84.45	86.53	84.74	85.63	76.3	77.73	75.13	76.41
VPN-Chat	73.84	72.56	70.67	71.60	72.33	70.55	70.25	70.40	66.51	64.45	70.45	67.32
VPN-Email	81.45	82.54	77.51	79.95	79.36	74.76	73.48	74.11	73.75	72.78	76.58	74.63
VPN-File Transfer	85.56	84.89	88.45	86.63	85.22	87.35	90.62	88.95	78.35	79.15	80.35	79.75
VPN-P2P	84.18	86.89	84.16	85.50	88.79	86.34	85.77	86.05	80.23	79.4	82.49	80.92
VPN-Streaming	84.09	85.29	81.92	83.57	81.05	85.74	84.91	85.32	78.37	77.21	79.31	78.25
VPN-VoIP	79.67	81.13	80.47	80.80	86.85	85.11	83.7	84.40	73.43	71.86	74.14	72.98

TABLE 10. The performance of the combined models in the CBS platform.

Type	Attention Bi-LSTM + 1D-CNN				Attention Bi-LSTM + 1D-CNN + SAE				Attention Bi-LSTM + 1D-CNN + SAE + GAN			
	Acc	Pr	Re	F1	Acc	Pr	Re	F1	Acc	Pr	Re	F1
Chat - (VPN)	85.4	86.52	88.7	87.6	90.4	91.16	91.5	91.33	99.86	99.60	99.6	99.6
Email - (VPN)	90.21	88.23	87.4	87.81	93.6	92.72	93.45	93.08	99.7	99.2	99.05	99.12
File Transfer - (VPN)	90.45	91.36	92.17	91.76	95.11	94.56	95.43	94.99	99.70	99.6	99.3	99.45
P2P - (VPN)	90.23	91.35	90.24	90.79	95.1	94.28	94.36	94.32	99.82	99.75	99.6	99.67
Streaming - (VPN)	89.13	91.2	90.42	90.81	94.2	94.36	95.11	94.73	99.75	99.65	99.3	99.47
VoIP - (VPN)	87.59	89.44	92.7	91.04	93.6	94.9	95.14	95.02	99.86	99.42	99.2	99.31

traffic into six-class categories for regular and encrypted traffic and 12-class traffic is 99.38%, 99.42%, and 99.3%, respectively.

Furthermore, accuracy, precision, and recall values for scenarios 2 through 4 demonstrate the model's effectiveness. As shown in Table 11, the weighted average of the F1 score is higher than 99.29%, indicating that our model extracts distinctive features from each class and performs efficiently. Table 12 is related to scenario 4 of the experiment, demonstrating that the proposed model excels at traffic classification. Table 12 displays CBS's traffic characterization performance as a weighted average. Additionally, as shown in Table 13, CBS can accurately identify the type of application. Despite the imbalance in the dataset, for example, email and chat classes have fewer instances than other types of traffic, CBS still performs well in categories with limited sample sizes thanks to SAE's ability to extract statistical features. The GAN model balances minority classes, enhancing the model's performance. Table 14 displays the accuracy, precision, recall, and F1 score values for the two-class classification scenario in Experiment 1, revealing that the CBS model achieved a high score for all metrics. Table 15 shows the accuracy, precision, recall, and F1 score values for Experiments 2 and 3.

C. MODEL COMPARISON FOR TRAFFIC CHARACTERIZATION RESULTS

This section compares the results obtained from the proposed model with those from other research. The comparison uses several metrics: recall, accuracy, precision, and the F1 score. However, since the studies presented in this section may not encompass all the experiments discussed in section V-C, some specific comparisons may only reflect some metrics. The effectiveness of the proposed model is compared with state-of-the-art approaches such as CSCNN [79], ICLSTM [63], C4.5 [49], and Deep Packet [8].

Table 16 presents the protocol-encapsulated traffic scenario's accuracy, recall, and F1 score metrics with two categories. The results indicate that the proposed model outperforms some of the compared approaches. According to Table 16, the CBS model performs better in precision, recall, and F1 metrics for Non-VPN traffic than the C4.5 model, with improvements of 9.16%, 10.87%, and 10.03%, respectively, confirming its superior efficiency. Similarly, in VPN traffic mode, these metrics demonstrate enhancements of 10.74% in precision, 7.69% in recall, and 9.22% in F1 score compared to C4.5. Table 17 displays a comparison of the CBS model's accuracy with 1D-CNN [57], ICLSTM [63], CSCNN [79], and attention-based LSTM (HAN) [78]. As reported in

TABLE 11. The performance of the CBS model in four experimental scenarios.

Experiment	Accuracy (Acc)	Precision (Pr)	Recall (Re)	F1 Score (F1)
1	99.82	99.75	99.68	99.71
2	99.45	99.44	99.32	99.38
3	99.38	99.47	99.37	99.42
4	99.7	99.38	99.22	99.3

TABLE 12. Traffic characterization results of the CBS model in Experiment 4.

Traffic Type	Acc	Pr	Re	F1
Chat	99.5	99.3	99.1	99.20
Email	99.7	98.4	99.3	98.85
File Transfer	99.75	99.62	99.6	99.80
P2P	99.66	99.59	99.53	99.56
Streaming	99.75	99.4	99	99.20
VoIP	99.7	99.2	98.1	98.65
VPN-Chat	99.69	99.61	99.55	99.58
VPN-Email	99.72	99.3	99.1	99.20
VPN-File Transfer	99.9	99.3	99.2	99.25
VPN-P2P	99.76	99.67	99.59	99.63
VPN-Streaming	99.62	99.58	99.55	99.63
VPN-VoIP	99.65	99.53	99.5	99.75
Weighted Average	**99.7**	**99.38**	**99.22**	**99.3**

Table 17, the proposed model accurately differentiates VPN traffic from Non-VPN traffic. The results indicate that the model presented in this paper is more precise than some state-of-the-art compared methods. The accuracy reported by [63] is 100%. In this work, overfitting may occur when the model learns the training data too well and cannot generalize to new data.

Consequently, the training data can be perfect, but the test data can be poor. In Table 16, 1D-CNN achieved 100% precision in the Non-VPN mode and 100% recall in the VPN mode. In Table 17, 1D-CNN [57] achieved 100% accuracy.

One possible explanation is overfitting, which occurs when a model learns the training data too well and cannot generalize to new data. Furthermore, the model may have been overfitted due to a lack of regularization techniques. After reviewing the evaluation metrics of all the compared models, it is evident that all two-class categories are relatively similar. Since there were only a few classes in the two-category classification scenario, all models could correctly identify

TABLE 13. Application identification of the CBS model (Experiment 4).

Application Identification	Acc	Pr	Re	F1
AIM-Chat	99.53	99.4	99.2	99.30
Email	99.43	99.26	99.11	99.18
Facebook	99.75	99.75	99.4	99.57
FTPS	99.77	99.7	99.65	99.67
Gmail	99.35	99.24	99.2	99.22
Hangouts	99.5	99.5	99.33	99.41
ICQ	99.75	99.63	99.45	99.54
Netflix	99.79	99.71	99.66	99.68
SCP	99.68	99.6	99.55	99.57
SFTP	99.72	99.65	99.58	99.61
Skype	99.84	99.84	99.5	99.67
Spotify	99.6	99.36	99.2	99
Torrent	99.7	99.65	99.62	99.63
Tor	99.66	99.58	99.54	99.56
VoipBuster	99.83	99.8	99.7	99.85
Vimeo	99.8	99.73	99.6	99.61
YouTube	99.68	99.68	99.25	99.46
Weighted Average	**99.67**	**99.59**	**99.44**	**99.51**

the distinguishing features between the two classes and performed similarly in evaluation metrics.

As shown in Table 18, the proposed CBS model compares with other models regarding precision, recall, and F1 score for both VPNs and Non-VPNs. In all of the tables in this paper, NaN stands for Not a Number, meaning the value of the compared metrics has yet to be stated. Experiments 2 and 3 of the proposed model correspond to regular and encrypted VPN traffic classification. In Experiment 2, the CBS model achieved precision, recall, and F1 score metrics of 99.4%, 99.32%, and 99.38%, respectively. In Experiment 3, these metrics values were 99.47%, 99.37%, and 99.42%, respectively. The high precision achieved by the CBS model indicates the effectiveness and superior performance of the proposed model in comparison to attention-based LSTM [78], 1D-CNN [57], ICLSTM [63], and C4.5 [49] studies.

Experiments 2 and 3 demonstrate that CBS outperformed 1D-CNN [57], HAN [78], C4.5 [49], and ICLSTM [63]. In Experiment 2, CBS showed a 13.94% increase in precision and a 13.52% increase in recall compared to 1D-CNN [57]. CBS also improved precision by 10.44% and recall by 13.82% compared to C4.5 [49]. Furthermore, in Experiment 3, the precision and recall of the proposed CBS model were higher by 4.58% and 2.07% compared to 1D-CNN [57] and by 15.47% and 11.77% compared to C4.5 [49]. Also, in Experiment 2, the proposed model improved the F1 score metric by 12.17% over C4.5 [49] and 13.73% over 1D-CNN [57].

Unlike the machine-learning method utilized by C4.5 [49], CBS automatically extracts features, which explains

TABLE 14. Protocol encapsulated traffic identification (Experiment 1).

Type	Acc	Pr	Re	F1
VPN	99.80	99.74	99.69	99.70
Non-VPN	99.84	99.76	99.67	99.72

why CBS outperforms the machine-learning method. Using human-crafted features in C4.5 results in traffic classification with low precision. On the other hand, 1D-CNN [57] employs spatial feature, which considers the relationship between packet bytes while disregarding the temporal features of the flow packet. Because the model does not incorporate statistical features to compensate for packet data trimmings, the 1D-CNN [57] model performs poorly in the six and 12-class categories. As shown in Table 18, in Experiment 2, the CBS model achieved higher accuracy than 1D-CNN [57] and C4.5 [49] by 17.65% and 10.45%, respectively. In Experiment 3, the proposed model outperforms ICLSTM [63]. According to Table 18, CBS accuracy in Experiment 3 is 12.38% better than C4.5 [49]. Similarly, the proposed CBS model achieves 14.35% and 6.48% accuracy improvements over HAN [78] in Experiments 2 and 3, respectively. Moreover, compared to ICLSTM [63], the proposed framework improves accuracy in Experiments 2 and 3.

The ICLSTM [63] model ignores essential statistical features in traffic classification. ICLSTM [63] utilizes the LSTM module to extract time-dependent packet features. However, Bi-LSTM modeling produces more accurate predictions than LSTM-based models. As a result, incorporating an attention layer into Bi-LSTM can improve the model performance and help make precise packet sequence predictions. The attention mechanism is more sensitive to packet bytes, significantly impacting traffic classification. In other words, the attention Bi-LSTM assigns a higher weight to the portion of the packet utilized to classify traffic flow. The CBS model outperforms ICLSTM because CBS incorporates the statistics feature and employs a Bi-LSTM attention model instead of LSTM. In [78], two models are used: Bi-LSTM and attention-based Bi-LSTM. This model classifies traffic exclusively based on the temporal relationship between packets. The models proposed in [78] do not consider the spatial relationship between packet bytes and session statistical features. Consequently, it has a lower performance than the CBS model. As shown in Table 19, the state-of-the-art compared models and the model proposed in this study achieved similar precision, recall, and F1 scores in Experiment 4. Since most papers on encrypted traffic classification rely on Experiment 4 for comparisons, this section compares the proposed CBS model with a more comprehensive list of existing methods.

In Experiment 4, a 12-category traffic classification experiment, our proposed approach achieved a performance of 99.38%, 99.22%, and 99.3% based on precision, recall, and F1 score metrics, respectively. By comparing the precision, recall, and F1 score metrics in Table 19, it is evident that the proposed model improves the performance of 1D-CNN [57] for Non-VPN classes by 13.45%, 13.21%, and 13.36%, respectively.

As depicted in Table 19, the CBS model outperforms 1D-CNN [57] in precision, recall, and F1 score metrics by 7.5%, 4.21%, and 5.94%, respectively, in VPN mode. Compared to the models proposed in [80] and [81], the CBS model performed exceptionally well. Compared to the C4.5 model in VPN mode, the proposed CBS model improves precision, recall, and F1 score metrics by 21.3%, 18.11%, and 19.79%, respectively. CBS improves F1 score, precision, and recall by 14.95%, 19.81%, and 17.49% in Non-VPN mode, compared to C4.5. The CBS model also improves the SAE model's precision, recall, and F1 metrics presented in the Deep Packet [8] by 12.56%, 10.31%, and 11.52% for Non-VPN classes, respectively. Additionally, the proposed CBS model outperformed the CNN model in the Deep Packet [8] by 10.45%, 10.31%, and 10.41% for each precision, recall, and F1 score metric in Non-VPN mode, respectively.

As shown in Table 19, we have achieved superior results by comparing our proposed method for classifying encrypted traffic with the methods presented in Experiment 4. Table 20 compares the CBS model's accuracy with state-of-the-art methods. As depicted in Table 20, the developed CBS model achieved 99.7% accuracy in Experiment 4. However, the models in ICLSTM [63], CSCNN [79], and [81] had 98.1%, 97.7%, and 98% accuracy, respectively. The accuracy of the two models in HAN [78] is 91.2% and 89.8%. Furthermore, 1D-CNN [57], FlowPic [59], and C4.5 [49] achieved 86.6%, 88%, and 90% accuracy, respectively. This paper presents a significantly more accurate model than all models evaluated in Table 20, demonstrating its efficiency.

CSCNN [79] focused only on the spatial features of the packets while disregarding statistical and temporal features. Additionally, this approach utilized only a limited number of CNN layers, which could hinder extracting complex features from packet data. Deep Packet [8] uses only two distinct models, CNN and SAE. It focused primarily on extracting the spatial features of the packet's bytes without considering the statistical and temporal features. It is possible to mistakenly classify several types of traffic without considering these distinguishing characteristics. Compared to the CBS model, Deep Packet's effectiveness can be diminished due to missing features (temporal and statistical).

In FlowPic [59], the CNN model prevented effective temporal and statistical features from extracting distinguishable features. This model exhibited inferior performance than CBS, as these features were not considered. The Datanet [19] employed three distinct networks, MLP, CNN, and SEA, relying only on spatial features between flow packet bytes. The Datanet MLP model performed notably poorly than other Datanet models due to its limited number of layers and neurons. These three models did not leverage statistical and temporal features to improve traffic classifier performance. These features are necessary to avoid significant information loss for traffic classification. Datanet performed less effectively due to these limitations.

TABLE 15. Regular and protocol encapsulated traffic classification.

Type	Experiment 2				Experiment 3			
	Acc	Pr	Re	F1	Acc	Pr	Re	F1
Chat	99.40	99.35	99.20	99.27	99.57	99.90	99.80	99.85
Email	99.15	99.10	99.00	99.05	99.10	99.20	99.10	99.15
File Transfer	99.66	99.90	99.55	99.72	99.25	99.40	99.30	99.35
P2P	99.74	99.72	99.63	99.67	99.71	99.66	99.53	99.59
Streaming	99.66	99.59	99.56	99.56	99.62	99.58	99.51	99.54
VoIP	99.10	99	99	99	99	99.10	99	99.05
Weighted Average	**99.45**	**99.44**	**99.32**	**99.38**	**99.38**	**99.47**	**99.37**	**99.42**

TABLE 16. The state-of-the-art methods comparison with the CBS model in Experiment 1.

Method	Non-VPN			VPN		
	Pr	Re	F1	Pr	Re	F1
C4.5 [49]	90.6	88.8	89.69	89	92	90.48
1D-CNN [57]	100	99.9	99.95	99.9	100	99.95
CSCNN [79]	99	98	98.50	97	99	97.99
ICLSTM [63]	100	100	100	99.9	100	99.95
Deep Packet-SAE [8]	98.84	100	99.92	100	99.59	99.79
Deep Packet-CNN [8]	99.77	99.92	99.84	99.58	98.75	99.16
CBS	**99.76**	**99.67**	**99.72**	**99.74**	**99.69**	**99.70**

TABLE 17. The state-of-the-art methods comparison with the CBS accuracy in two-category scenario.

Method	Acc
1D-CNN [57]	99.99
ICLSTM [63]	100
CSCNN [79]	98
HAN [78]	99.5
Bi-LSTM [78]	99.7
CBS	**99.82**

According to the proposed model in [80], the F1 score values for Non-VPN and VPN modes are 87.45% and 96.29%, respectively. Due to the lack of statistical and temporal features, this model does not perform as well as our proposed model. This model only relies on spatial features. Furthermore, [81] reported 97.66% and 98.66% F1 scores for Non-VPN and VPN modes, respectively. The authors of [81] ignored the importance of temporal features in traffic classification, which significantly influences the diversity of traffic types. Neglecting this feature can hurt traffic classification. Due to its lack of temporal features, this approach is less practical than CBS. An effective traffic classification relies on consideration of temporal features, such as timing and sequencing.

In SpCaps [58], the F1 score value is 93.3% when combined with both Non-VPN and VPN modes. Statistics and temporal features, essential for classifying traffic, are not included in [58]. Consequently, it cannot accurately extract significant features from input data and, hence, cannot outperform CBS. On the other hand, the model evaluated in [45] reported an F1 score of 97.59% for the Non-VPN mode. However, [45] uses only spatial features to classify traffic using a CNN model. Notably, a significant portion of distinguishing information is lost by disregarding traffic data's statistical and temporal features. Therefore, this limitation makes the CBS model less effective.

D. MODEL COMPARISON FOR APPLICATION IDENTIFICATION RESULTS

The assessment of the proposed model's performance involves various criteria, including the application identification factor that considers the end user's application, such as Facebook and ICQ. The CBS application identification results are compared to other studies in Table 21. In Table 21, the proposed approach achieved outstanding performances with accuracy, precision, recall, and F1 score of 99.67%, 99.59%, 99.44%, and 99.51%, respectively. The comparison with other models, such as Deep Packet [8], Datanet [19], and CSCNN [79], showed that CBS outperformed them in all metrics. Furthermore, the application identification results of the proposed CBS model indicated that the features extracted from the training dataset were comprehensive enough to capture each application's distinctive features. In comparison, the Deep Packet [8] model with CNN and SAE models achieved F1 scores of 95% and 98%, respectively, for application identification. Similarly, CSCNN [79] reported an F1 score of 96.3%, while Datanet [19] reached 94.1%.

E. COMPARISON OF ENCRYPTED TRAFFIC CLASSIFICATION AND APPLICATION IDENTIFICATION

The CBS traffic characterization and application identification confusion matrix are shown in Fig. 16 and Fig. 17. Training CBS to precisely classify traffic sessions, such as web browsing, email, file transfer, and video streaming, is achievable by employing spatial, temporal, and statistical features. The matrix rows correspond to each class's actual instances, while the columns represent the predicted label. The matrix is normalized to ensure row-level normalization.

TABLE 18. Non-VPN and VPN state-of-the-art methods were compared with the CBS model in Experiments 2 & 3.

Method	Experiment 2				Experiment 3			
	Acc	Pr	Re	F1	Acc	Pr	Re	F1
C4.5 [49]	89	89	85.5	87.21	87	84	87.6	85.76
1D-CNN [57]	81.8	85.5	85.8	85.65	98.6	94.9	97.3	96.09
ICLSTM [63]	98.2	99.1	99.4	99.25	98.7	98.3	97.6	97.95
HAN [78]	85.1	NaN	NaN	NaN	92.9	NaN	NaN	NaN
Bi-LSTM [78]	89.3	NaN	NaN	NaN	94.8	NaN	NaN	NaN
CBS	**99.45**	**99.44**	**99.32**	**99.38**	**99.38**	**99.47**	**99.37**	**99.42**

TABLE 19. The state-of-the-art methods were compared with the CBS model in Experiment 4.

Method	Non-VPN			VPN			Non-VPN + VPN		
	Pr	Re	F1	Pr	Re	F1	Pr	Re	F1
C4.5 [49]	84.3	79.3	81.72	78.2	81.3	79.72	NaN	NaN	NaN
1D-CNN [57]	85.8	85.9	85.85	92	95.2	93.57	NaN	NaN	NaN
CSCNN [79]	97.4	97.4	97.40	97	99	97.99	97.4	97.4	97.4
ICLSTM [63]	96.6	97.6	97.00	98.5	97.8	98.00	98	98	98.1
Deep Packet-SAE [8]	86.6	88.8	87.69	99.1	99.1	99.10	92	92	92
Deep Packet-CNN [8]	88.8	88.8	88.8	99.58	98.75	99.16	94	93	93
SPCaps [58]	98.6	98.68	98.63	99.9	99.9	99.83	93.3	93.3	93.3
Text-CNN [80]	87.6	87.3	87.45	95.2	97.4	96.29	NaN	NaN	NaN
CNN-SAE-Based [81]	97.1	97.83	97.66	98.66	99.1	98.66	98	98	98
Datanet-SAE [19]	NaN	NaN	NaN	NaN	NaN	NaN	98.83	98.81	98.82
Image-Based [45]	97.51	97.67	97.59	99.73	98.82	99.26	98.65	98.64	98.64
CBS	**99.25**	**99.11**	**99.21**	**99.5**	**99.41**	**99.51**	**99.38**	**99.22**	**99.3**

The main diagonal of the confusion matrix contains elements that represent correctly classified samples, while other regions have values that correspond to incorrectly classified samples. As illustrated in Fig. 16 and 17, the dark-colored elements on the major diagonal of the confusion matrix demonstrate CBS's capability to effectively and accurately classify traffic characterization and application identification tasks. Invisible payloads pose a challenge to traditional classification methods for encrypted traffic. However, patterns can be identified by analyzing features such as packet size and timing, session duration, and inter-packet timing to classify traffic. For example, by analyzing the inter-packet timings and session duration, it may be possible to distinguish between different types of encrypted traffic, such as VPN or SSL traffic. Therefore, using spatial, temporal, and statistical features can provide additional insights into the types of traffic on the network and help identify applications. Multiple features enable CBS to achieve a high accuracy rate despite encrypted traffic. Employing multiple features can reduce false positive and false negative rates, leading to more accurate classification results.

The confusion matrix of application identification in Fig. 16 and traffic characterization in Fig. 17 is light-colored since the CBS model accurately distinguishes traffic types and applications. The confusion matrix shows that the model correctly classified most traffic, as indicated by the dark diagonal elements. As can be seen in the confusion matrix, the non-diagonal elements are incredibly light, meaning the model rarely misclassifies traffic. The classification model is trained using cross-validation. This technique prevents the overfitting of the model to the training data. Therefore, the model can be generalized well to new traffic samples. Cross-validation can help achieve a light-colored confusion matrix in encrypted traffic classification by ensuring the model does not overfit the training data. In addition to preventing overfitting, cross-validation can also help improve the model's accuracy. This is because cross-validation allows the model to be trained in a wider variety of data, which can help the model to learn more generalizable features.

Analyzing the confusion matrix can provide insight into the strengths and weaknesses of the CBS deep learning model for encrypted traffic classification. For example, suppose the model makes many false positives. In that case, the precision will be low, indicating that the model incorrectly classified numerous instances, and non-diagonal elements will have a darker color. Similarly, if a model fails to identify many positive instances correctly, its recall score will be low. Using the GAN network to balance data allowed CBS to effectively identify the type of email application traffic with a relatively small sample size, as depicted by the confusion matrix in Fig. 16.

TABLE 20. The accuracy of state-of-the-art methods compared with the CBS model in Experiment 4.

Method	Acc (Non-VPN + VPN)
ICLSTM [63]	98.1
CSCNN [79]	97.7
SPCaps [58]	99.1
1D-CNN [57]	86.6
HAN [78]	89.5
Bi-LSTM [78]	91.2
CNN-SAE-Based [81]	98
Flowpic [59]	88
CBS	**99.7**

The non-diagonal elements of email application traffic are represented by very light-colored elements, reflecting CBS's precise classification of email application traffic. Similarly, the same analysis applies to the email traffic type depicted in Fig. 17. According to Fig. 17, non-diagonal elements demonstrate CBS' accuracy in detecting this type of traffic due to the simultaneous use of multiple spatial, temporal, and statistical features and the GAN network. We mentioned email application traffic and email traffic because both samples were imbalanced. The objective was to demonstrate how CBS could effectively classify imbalanced traffic using a few samples.

F. TRAINING ANALYSIS

Early convergence in the CBS model is achieved when the model has learned all discriminative patterns in the data and does not require additional training epochs. This is crucial in encrypted traffic classification, as it stabilizes the model's performance without significantly improving accuracy. The number of epochs needed for convergence depends on the model's complexity, dataset size and quality, and selected hyperparameters. Increasing the number of epochs can improve model accuracy to a certain point, but beyond this point, it can lead to overfitting. The number of epochs can be adjusted during training to optimize accuracy and prevent overfitting by monitoring the model's performance on the validation set. Data preprocessing techniques can significantly impact the relationship between convergence and epoch in deep-learning-based encrypted traffic classification. The most significant reasons for CBS's rapid convergence are as follows.

- **Preprocessing:**The convergence of models depends heavily on data preprocessing. The convergence speed of a model can be enhanced with proper preprocessing techniques, especially if those techniques are aligned with the characteristics of the data. CBS considers comprehensive preprocessing before training.
- **Feature Scaling and Normalization:** Feature Scaling can help the model converge faster by ensuring that updates to the model's parameters are more consistent across features. CBS scales each byte in the packet from [0-255] to [0-1]. CBS employs the normalization method outlined in [77]. Using batch normalization drastically reduces the epochs required to train deep neural networks. Data normalization prevents overfitting by eliminating outliers. Outliers are extreme values that differ from most data samples in a dataset.
- **Feature Selection:** The model can converge faster and prevent overfitting by reducing features. CBS simultaneously uses several spatial, temporal, and statistical features as a complete feature set.
- **Data Augmentation:** CBS uses GAN networks for data augmentation and balancing to increase the size of the training dataset, enabling the model to learn and generalize patterns through diverse examples. This robust training process stabilizes the learning process and may lead to earlier convergence. A balanced dataset ensures the model encounters various traffic patterns without bias, resulting in successful learning of underlying representations and patterns.
- **Well-structure Dataset:** Complex, noisy, or unstructured datasets can impede the early convergence of encrypted traffic datasets, causing more training epochs in the training process. The CBS model converges faster due to its ability to identify discriminative features in the well-structured ISCX VPN-Non VPN 2016 dataset. Preprocessing techniques can enhance the training data quality, making it easier for the model to learn underlying patterns.

TABLE 21. The state-of-the-art methods compared with application identification of the CBS model.

Application Identification	Acc	Pr	Re	F1
Deep Packet-SAE [8]	NaN	95	96	95
Deep Packet-CNN [8]	98	98	98	98
CSCNN [79]	97.9	97.9	98.6	96.3
Datanet [19]	96.1	94.4	93.9	94.1
CBS	**99.67**	**99.59**	**99.44**	**99.51**

CBS has achieved higher model accuracy and faster convergence for encrypted traffic classification through effective preprocessing. Figs. 18(a)-(c) provide insight into the proposed model's convergence. By analyzing the proposed CBS model more precisely, we explore the reasons for better performance gains. Figs. 18(a) and 18(b) show the training accuracy, validation accuracy, training loss, and validation loss parameters at the end of each epoch. As illustrated in Figs. 18(a) and 18(b), the proposed model's average accuracy for training and validation has converged well. CBS gains valuable insights into the model's learning process by feature engineering and feature extraction. Additionally, the accuracy of the proposed model is compared with other models based on the number of epochs. As shown in Fig. 18(a), our proposed model allows the training model to converge in minimal epochs. Fig. 18(a) shows that our model requires

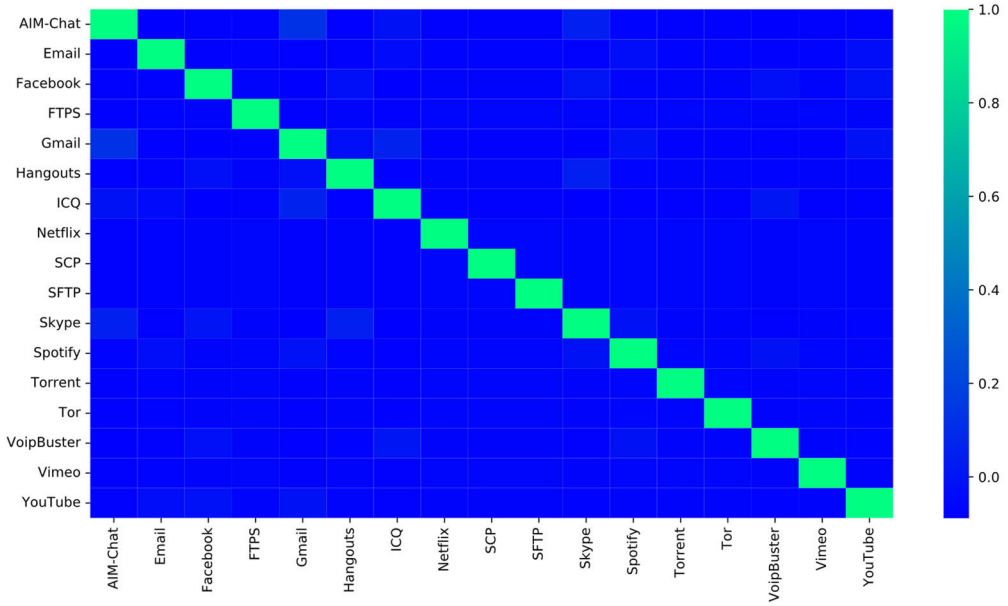


FIGURE 16. The confusion matrix for application identification.

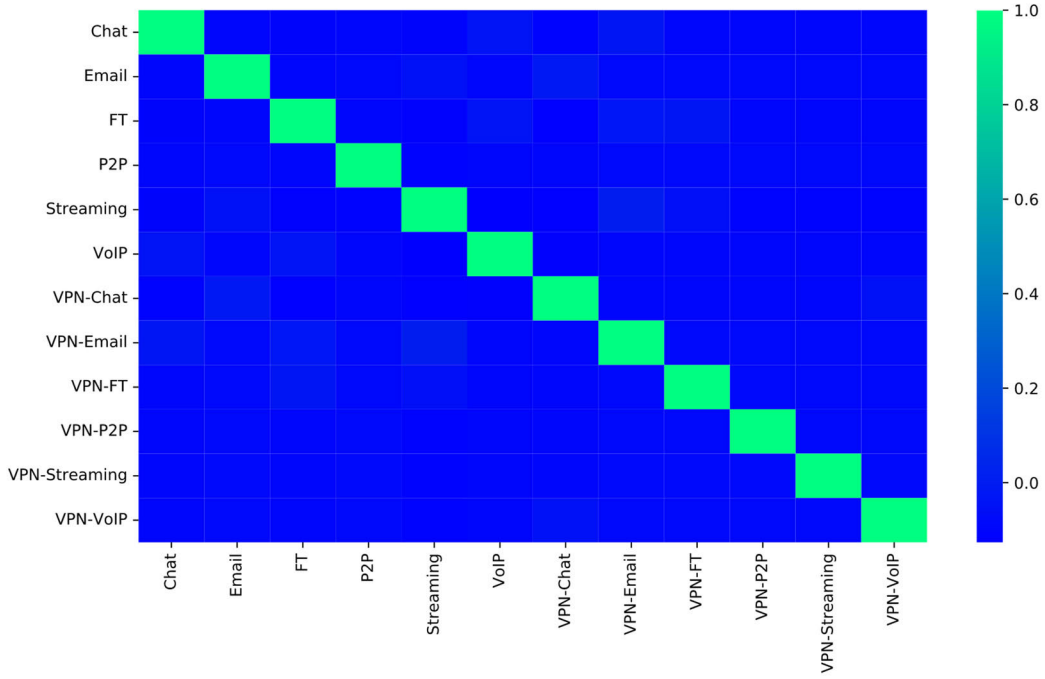


FIGURE 17. The confusion matrix for traffic characterization.

only four converging epochs. Fig. 18(c) also reveals that our proposed model achieved 95% accuracy in four epochs and converged in fewer epochs than other studied models. For all experiments, the initial epoch value equals 50; however, in Fig. 18, it only shows up to 20. During training, a saturation point occurs in CBS when loss and accuracy values no longer improve significantly. Since training accuracy and loss have stopped changing after epoch 20, we don't consider epochs after 20 to 50. For several reasons, using simultaneous spatial, temporal, and statistical features in encrypted traffic classification can improve CBS convergence.

- Combining multiple features can result in a broader and more diverse range of inputs for the deep learning model, enabling it to discern complex patterns and relationships in the data. This can facilitate rapid model convergence.

- The risk of overfitting can be reduced by combining spatial, temporal, and statistical features. When the model becomes too specialized for training data, it performs poorly on new data. This forces the model to generalize and acquire more robust data representations.
- Using multiple features in deep learning algorithms can prevent the model from getting stuck in suboptimal solutions, a common issue. This is because local minima can cause poor convergence and performance. Using multiple features reduces this risk.
- The deep learning model can detect nuanced distinctions in traffic by combining multiple features. For instance, different types of encrypted traffic may have similar spatial features but differ significantly in temporal or statistical aspects. Integrating these features allows the model to understand traffic comprehensively and converge rapidly.

The graphs in Fig. 18 have a steady slope. The following explanations can justify this behavior. During the training of the CBS model for encrypted traffic classification, a steady decrease in training loss and an increase in training accuracy can indicate a consistent and stable learning process. Training loss and accuracy can be steadily sloped if the training dataset is well-balanced and represents real-world encrypted traffic distribution. As CBS's architecture captures relevant features and its model is expressive, training loss and accuracy will likely follow a steady slope. A steady slope can also be achieved by early stopping. Early stopping stops the training process when the validation loss increases during training. Classification tasks often use a cross-entropy loss function. Training loss and accuracy can be affected by the choice of loss function. Cross-entropy loss is helpful in training because it encourages more accurate predictions. It guides the optimization process during training, allowing the model to converge towards superior results. A steady slope in training can be achieved with an appropriate loss function aligned with the model's encrypted traffic classification objectives. By processing the data appropriately, training loss and accuracy can be steadily reduced.

G. RUNTIME ANALYSIS

The runtime of 1D-CNN, Bi-LSTM, FC, and SAE models for encrypted traffic classification based on the ISCX VPN-Non VPN 2016 dataset depends on several factors, including:

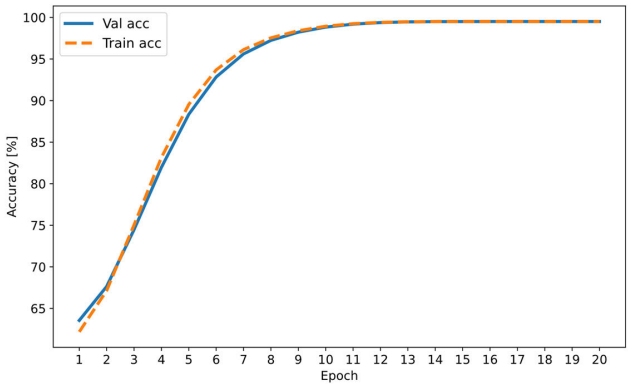
- Configuration of the system's hardware and software for training and deploying the classification model.
- Classification model complexity and size.
- The quantity of traffic to be classified.

1D-CNN models are faster than Bi-LSTM and SAE models due to their more straightforward structure and parallel processing capabilities. The runtime of 1D-CNN models depends on the number and size of convolutional layers, network architecture complexity, and input data size. Parallel processing capabilities embedded in 1D-CNNs can reduce computational costs, mainly when dealing with large datasets. The number of LSTM layers, network architecture complexity, and input sequence length influence Bi-LSTM runtime. Bi-LSTMs, due to their sequential nature, perform worse computationally than 1D-CNNs, as they process the input sequence one step at a time. Stacking autoencoders (SAEs) are unsupervised deep learning models that can learn features and reduce dimensionality without labeled data, making them ideal for extracting meaningful representations from input data. Their runtime depends on layer size, number of neurons, and input and output vector size. The training algorithm also plays a crucial role in SAE's execution, as it requires multiple forward and backward passes for representation learning, making them train slower than 1D-CNNs. 1D-CNNs are ideal for classification tasks due to their parallel processing capabilities, particularly with GPUs. SAEs have a more prolonged training phase due to iterative optimization, while Bi-LSTM networks can be trained quickly using parallelization techniques. Due to their simpler structure, 1D-CNNs and FC networks are more straightforward to parallelize. Fully connected networks (FC) are the fastest and most efficient due to their simplicity, fewer parameters, and simultaneous computations.

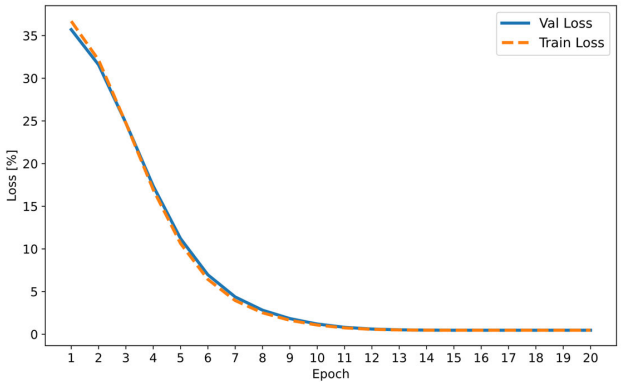
Generative adversarial networks (GANs) are slower than other CBS models like 1D-CNNs, attention-based LSTMs, stacked autoencoders (SAEs), and fully connected networks (FCNs) due to the complexity of network architectures and the need for convergence. GANs require multiple iterations and complex computations to optimize two networks simultaneously, leading to higher computational demands and longer training times than other CBS models. Memory requirements for storing intermediate activations, gradients, and generated samples also affect GAN runtime. A limited amount of memory may cause frequent data transfers between memory and slower storage devices, further slowing down training. Table 22 shows the average execution time for the GAN, FC, 1D-CNN, Attention Bi-LSTM, and SAE models based on the ISCX VPN-Non VPN 2016 dataset. According to Table 22, the FC model has the lowest average execution time due to the abovementioned factors. The 1D-CNN model performs better than other models except FC. The average execution time is considered for each model in the feature extraction phase. Since this step is very time-consuming, each model's performance should be evaluated regarding how long it takes to complete.

H. MEMORY ANALYSIS

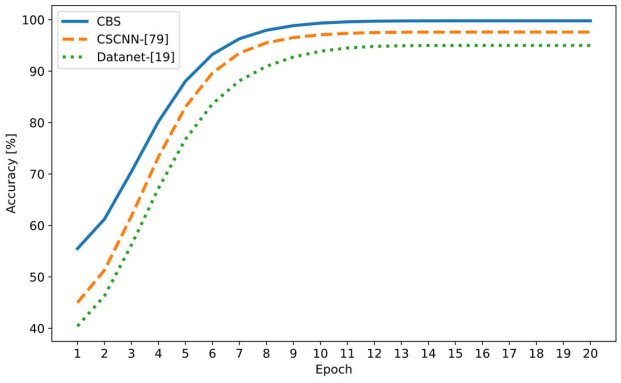
Training and inference processes require storing each parameter in memory, so the number of parameters can affect memory usage for 1D-CNN model. In 1D-CNN architectures, factors like filters, convolutional layer size, stride size, pooling filter size, and fully connected layers influence memory usage. The total memory usage for each layer can be calculated by summing up the memory requirements for each layer. Bi-LSTM's memory requirement depends on the size and number of LSTM units in each direction. Autoencoders, consisting of an encoder and decoder, have memory require-



(a) Training and validation progress accuracy based on the number of epochs.



(b) Training and validation progress loss based on the number of epochs.



(c) Training accuracy comparison based on the number of epochs.

FIGURE 18. Accuracy-Loss changes during training of the CBS model and compared models.

ments influenced by input and output vector sizes, layers, and neurons. Due to complex architectures and multiple layers, 1D-CNNs and Bi-LSTMs have more parameters, requiring higher memory requirements than SAEs. 1D-CNNs have more layers and filters, requiring additional weights and activation storage. Bi-LSTMs have a more straightforward structure with fewer parameters, resulting in lower memory usage. SAE uses compact representations with a more transparent structure, requiring less memory than 1D-CNNs. 1D-CNNs have more layers, larger filters, and fully connected layers, which need more memory for weights and activations. Direct connections between neurons in a fully connected network require more memory. As the number of neurons increases, the number of parameters quadratically increases, requiring a parameter for each connection between two neurons.

TABLE 22. Runtime execution time for used models in the CBS.

Model Runtime (S)				
GAN	FC	1D-CNN	Attention Bi-LSTM	SAE
23700	9000	11520	14160	15060

TABLE 23. Memory consumption for used models in the CBS.

Model Memory Usage (MB)				
GAN	FC	1D-CNN	Attention Bi-LSTM	SAE
2430	1200	900	650	580

1D-CNNs have sparse connectivity patterns, with each neuron having only partial connections to the next layer. 1D-CNNs use local connectivity and shared weights in convolutional layers, reducing memory usage. However, Bi-LSTMs use recurrent connections and have a more complex structure, requiring more memory than SAEs. They also learn long-term dependencies using recurrent connections, making them memory intensive. Unlike Bi-LSTMs, SAEs are feedforward neural networks without recurrent connections, making them less memory-intensive. They focus on learning compact representations of input data using less memory than fully connected networks. SAEs have fewer layers, weights, and activations, allowing them to train efficient models from input data.

GANs, which train both the generator and discriminator simultaneously, have higher memory requirements due to their simultaneous storage of intermediate activation values, gradients, and generated samples. In contrast, 1D-CNNs, attention Bi-LSTMs, SAEs, and Fully Connected Networks have simpler architectures and require less memory storage. GANs have many layers and parameters, which need more memory for storing their weights, kernel filters, stride, and gradients. Table 23 shows the memory used by GAN, FC, 1D-CNN, Attention Bi-LSTM, and SAE models based on the ISCX VPN-Non VPN 2016 dataset. As shown in Table 23, the FC model has the second-highest memory usage for the abovementioned reasons. The SAE model outperforms other models because it has fewer weights and neuron layers. Because feature extraction is a memory-intensive phase, memory consumption is measured based on how much memory each model consumes during this phase.

VII. DISCUSSION

The proposed model was adapted to address the trimming packet data during the preprocessing phase, which can lead

to the loss of valuable information. Statistical features like inter-arrival time and packet length were used to compensate for this loss. The model was also implemented to capture the spatial features of network traffic packets due to their sequential and time-series nature. According to [57], 2D-CNN converts data packet bytes into a two-dimensional matrix, but its sequential nature may reduce model efficiency. Bytes from two distinct rows are independent, affecting feature extraction. The proposed CBS model should address more than just packet spatial and statistical features for efficient performance. CBS uses attention-based Bi-LSTM to improve performance evaluation metrics like recall, precision, and F1 score by identifying unique patterns between packets. This approach overcomes the limitations of LSTM in learning long sequence dependencies and enhances the model's accuracy. CBS discovers short-term and long-term packet dependencies using attention-based Bi-LSTM, which assigns more weight to packets with a higher impact on classifying various application types or traffic classes. The attention mechanism enhances detection efficiency. CBS aims to extract the most efficient features in traffic classification from multiple aspects and learns to extract a comprehensive feature set to improve the model's efficiency. The study found that the dataset used in the analysis was imbalanced, with varying sample numbers among classes, reducing efficiency in traffic classification. The GAN network mitigated this issue, improving traffic classification efficiency. The GAN model added similar instances to minority classes and initiated feature extraction training. However, removing the GAN model from the primary CBS platform could negatively impact evaluation metrics and system performance. CBS is designed for offline scenarios and not applicable to online scenarios, making it unsuitable for real-time input data analysis or classification. Therefore, alternative platforms are needed for real-time data stream analysis. Modifying input data before using the CBS model for real-time traffic classification is necessary. While CBS utilizes pre-stored data, real-time applications require streaming data. Statistical features in the proposed model cannot be used in real-time because they are based on a complete working session. In real-time scenarios, the session's beginning can be detected, but its end can only be determined once it ends. Due to this uncertainty, many statistical features cannot be extracted.

VIII. CONCLUSION AND FUTURE WORK

Traffic classification is crucial in network management, especially with the increasing use of encrypted traffic. Spatial or temporal feature extraction methods often fail to detect certain statistical features, leading to information loss. This paper proposes a comprehensive feature set platform called CBS that combines all statistical, spatial, and temporal features extracted from traffic files for encrypted traffic classification. The architecture employs 1D-CNN, attention-based Bi-LSTM, and SAE models. Combining features improves model performance by capturing different aspects of input data and leveraging each feature's potential for more accurate predictions. The proposed model uses the GAN network to generate synthetic samples for imbalanced classes to address dataset imbalance. GANs can prevent overfitting by increasing the dataset size and developing new data samples. Experiments and comparisons show high accuracy, precision, recall, and F1 score for the proposed traffic characterization model. In the future, we will explore encrypted traffic classification in the real world. A solution for classifying stream data in the real world will be developed.

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.

REFERENCES

[1] J. Krupski, W. Graniszewski, and M. Iwanowski, "Data transformation schemes for CNN-based network traffic analysis: A survey," *Electronics*, vol. 10, no. 16, p. 2042, Aug. 2021.

[2] Z. Bu, B. Zhou, P. Cheng, K. Zhang, and Z.-H. Ling, "Encrypted network traffic classification using deep and parallel network-in-network models," *IEEE Access*, vol. 8, pp. 132950–132959, 2020.

[3] J. Cheng, Y. Wu, E. Yuepeng, J. You, T. Li, H. Li, and J. Ge, "MATEC: A lightweight neural network for online encrypted traffic classification," *Comput. Netw.*, vol. 199, Nov. 2021, Art. no. 108472.

[4] Z. Zou, J. Ge, H. Zheng, Y. Wu, C. Han, and Z. Yao, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *Proc. IEEE 20th Int. Conf. High Perform. Comput. Commun., IEEE 16th Int. Conf. Smart City, IEEE 4th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Jun. 2018, pp. 329–334.

[5] A. Azab, M. Khasawneh, S. Alrabaee, K.-K.-R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges," *Digit. Commun. Netw.*, 2022, doi: 10.1016/j.dcan.2022.09.009.

[6] M. S. Sheikh and Y. Peng, "Procedures, criteria, and machine learning techniques for network traffic classification: A survey," *IEEE Access*, vol. 10, pp. 61135–61158, 2022.

[7] A. I. Getman and M. K. Ikonnikova, "A survey of network traffic classification," *Proc. Inst. Syst. Program. RAS*, vol. 32, no. 6, pp. 137–154, 2021.

[8] M. Lotfollahi, M. J. Siavoshani, R. S. H. Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning," *Soft Comput.*, vol. 24, no. 3, pp. 1999–2012, Feb. 2020.

[9] C. Dong, C. Zhang, Z. Lu, B. Liu, and B. Jiang, "CETAnalytics: Comprehensive effective traffic information analytics for encrypted traffic classification," *Comput. Netw.*, vol. 176, Jul. 2020, Art. no. 107258.

[10] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: An overview," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 76–81, May 2019.

[11] M. Shen, K. Ye, X. Liu, L. Zhu, J. Kang, S. Yu, Q. Li, and K. Xu, "Machine learning-powered encrypted network traffic analysis: A comprehensive survey," *IEEE Commun. Surveys Tuts.*, vol. 25, no. 1, pp. 791–824, 1st Quart., 2023.

[12] A. Agrawal, A. Bhatia, A. Bahuguna, K. Tiwari, K. Haribabu, D. Vishwakarma, and R. Kaushik, "A survey on analyzing encrypted network traffic of mobile devices," *Int. J. Inf. Secur.*, vol. 21, no. 4, pp. 873–915, Aug. 2022.

[13] P. Wang, X. Chen, F. Ye, and Z. Sun, "A survey of techniques for mobile service encrypted traffic classification using deep learning," *IEEE Access*, vol. 7, pp. 54024–54033, 2019.

[14] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: Mobile encrypted traffic classification using multimodal deep learning," *Comput. Netw.*, vol. 165, Dec. 2019, Art. no. 106944.

[15] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: Encrypted traffic classification via multimodal multitask deep learning," *J. Netw. Comput. Appl.*, vols. 183–184, Jun. 2021, Art. no. 102985.

[16] B. Sun, W. Yang, M. Yan, D. Wu, Y. Zhu, and Z. Bai, "An encrypted traffic classification method combining graph convolutional network and autoencoder," in *Proc. IEEE 39th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Nov. 2020, pp. 1–8.

[17] E. Papadogiannaki and S. Ioannidis, "A survey on encrypted network traffic analysis applications, techniques, and countermeasures," *ACM Comput. Surv.*, vol. 54, no. 6, pp. 1–35, Jul. 2022.

[18] A. Shahraki, M. Abbasi, A. Taherkordi, and A. D. Jurcut, "Active learning for network traffic classification: A technical study," *IEEE Trans. Cogn. Commun. Netw.*, vol. 8, no. 1, pp. 422–439, Mar. 2022.

[19] P. Wang, F. Ye, X. Chen, and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, vol. 6, pp. 55380–55391, 2018.

[20] R. Li, X. Xiao, S. Ni, H. Zheng, and S. Xia, "Byte segment neural network for network traffic classification," in *Proc. IEEE/ACM 26th Int. Symp. Quality Service (IWQoS)*, Jun. 2018, pp. 1–10.

[21] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning," in *Proc. Netw. Traffic Meas. Anal. Conf. (TMA)*, Jun. 2018, pp. 1–8.

[22] S. Wang, J. Cao, and P. S. Yu, "Deep learning for spatio-temporal data mining: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 8, pp. 3681–3700, Aug. 2022.

[23] Y. Zeng, H. Gu, W. Wei, and Y. Guo, "Deep-Full-Range: A deep learning based network encrypted traffic classification and intrusion detection framework," *IEEE Access*, vol. 7, pp. 45182–45190, 2019.

[24] M. Camelo, P. Soto, and S. Latré, "A general approach for traffic classification in wireless networks using deep learning," *IEEE Trans. Netw. Service Manage.*, vol. 19, no. 4, pp. 5044–5063, Dec. 2022.

[25] L. Yu, J. Dong, L. Chen, M. Li, B. Xu, Z. Li, L. Qiao, L. Liu, B. Zhao, and C. Zhang, "PBCNN: Packet bytes-based convolutional neural network for network intrusion detection," *Comput. Netw.*, vol. 194, Jul. 2021, Art. no. 108117.

[26] A. Aggarwal, M. Mittal, and G. Battineni, "Generative adversarial network: An overview of theory and applications," *Int. J. Inf. Manage. Data Insights*, vol. 1, no. 1, Apr. 2021, Art. no. 100004.

[27] H. Navidan, P. F. Moshiri, M. Nabati, R. Shahbazian, S. A. Ghorashi, V. Shah-Mansouri, and D. Windridge, "Generative adversarial networks (GANs) in networking: A comprehensive survey & evaluation," *Comput. Netw.*, vol. 194, Jul. 2021, Art. no. 108149.

[28] K. I. D. Alsulami, "Application-based network traffic generator for networking AI model development," Ph.D. dissertation, Univ. Dayton, 2021. [Online]. Available: http://rave.ohiolink.edu/etdc/view?acc_num=dayton1619387614152354

[29] Z. Zhang, L. Liu, X. Lu, Z. Yan, and H. Li, "Encrypted network traffic classification: A data driven approach," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2020, pp. 706–712.

[30] W. Wei, H. Gu, W. Deng, Z. Xiao, and X. Ren, "ABL-TC: A lightweight design for network traffic classification empowered by deep learning," *Neurocomputing*, vol. 489, pp. 333–344, Jun. 2022.

[31] J. Yan and J. Yuan, "A survey of traffic classification in software defined networks," in *Proc. 1st IEEE Int. Conf. Hot Inf.-Centric Netw. (HotICN)*, Aug. 2018, pp. 200–206.

[32] M. A. I. M. Aminuddin, Z. Fitri, M. Kaur, and D. Singh, "A survey on Tor encrypted traffic monitoring," *Int. J. Adv. Comput. Sci. Appl.*, vol. 9, no. 8, 2018, doi: 10.14569/IJACSA.2018.090815.

[33] A. M. Sadeghzadeh, S. Shiravi, and R. Jalili, "Adversarial network traffic: Towards evaluating the robustness of deep-learning-based network traffic classification," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1962–1976, Jun. 2021.

[34] G. D'Angelo and F. Palmieri, "Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction," *J. Netw. Comput. Appl.*, vol. 173, Jan. 2021, Art. no. 102890.

[35] U. Prabu and V. Geetha, "Self-organizing deep learning model for network traffic classification," in *Proc. Inventive Commun. Comput. Technol. (ICICCT)*. Singapore: Springer, 2022, pp. 419–425.

[36] O. Salman, I. H. Elhajj, A. Kayssi, and A. Chehab, "A review on machine learning–based approaches for Internet traffic classification," *Ann. Telecommun.*, vol. 75, nos. 11–12, pp. 673–710, Dec. 2020.

[37] A. Boumhand, K. Singh, Y. Hadjadj-Aoul, M. Liewig, and C. Viho, "Network traffic classification for detecting multi-activity situations," in *Proc. IEEE Symp. Comput. Commun. (ISCC)*, Jul. 2023, pp. 681–687.

[38] A. Elngar and A. Burlea-Schiopoiu, "Feature selection and dynamic network traffic congestion classification based on machine learning for Internet of Things," *Wasit J. Comput. Math. Sci.*, vol. 2, no. 2, pp. 76–91, Jul. 2023.

[39] C.-Y. Shin, J.-T. Park, U.-J. Baek, and M.-S. Kim, "A feasible and explainable network traffic classifier utilizing DistilBERT," *IEEE Access*, vol. 11, pp. 70216–70237, 2023.

[40] J. Dai, X. Xu, and F. Xiao, "GLADS: A global–local attention data selection model for multimodal multitask encrypted traffic classification of IoT," *Comput. Netw.*, vol. 225, Apr. 2023, Art. no. 109652.

[41] K.-C. Chiu, C.-C. Liu, and L.-D. Chou, "CAPC: Packet-based network service classifier with convolutional autoencoder," *IEEE Access*, vol. 8, pp. 218081–218094, 2020.

[42] X. Wang, S. Chen, and J. Su, "Automatic mobile app identification from encrypted traffic with hybrid neural networks," *IEEE Access*, vol. 8, pp. 182065–182077, 2020.

[43] B. Yang and D. Liu, "Research on network traffic identification based on machine learning and deep packet inspection," in *Proc. IEEE 3rd Inf. Technol., Netw., Electron. Autom. Control Conf. (ITNEC)*, Mar. 2019, pp. 1887–1891.

[44] J. Zhao, X. Jing, Z. Yan, and W. Pedrycz, "Network traffic classification for data fusion: A survey," *Inf. Fusion*, vol. 72, pp. 22–47, Aug. 2021.

[45] Y. He and W. Li, "Image-based encrypted traffic classification with convolution neural networks," in *Proc. IEEE 5th Int. Conf. Data Sci. Cyberspace (DSC)*, Jul. 2020, pp. 271–278.

[46] S. Dong and R. Jain, "RETRACTED: Flow online identification method for the encrypted skype," *J. Netw. Comput. Appl.*, vol. 132, pp. 75–85, Apr. 2019.

[47] G. Sun, T. Chen, Y. Su, and C. Li, "Internet traffic classification based on incremental support vector machines," *Mobile Netw. Appl.*, vol. 23, no. 4, pp. 789–796, Aug. 2018.

[48] A. S. Da Silva, C. C. Machado, R. V. Bisol, L. Z. Granville, and A. Schaeffer-Filho, "Identification and selection of flow features for accurate traffic classification in SDN," in *Proc. IEEE 14th Int. Symp. Netw. Comput. Appl.*, Sep. 2015, pp. 134–141.

[49] G. Draper-Gil, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and VPN traffic using time-related features," in *Proc. 2nd Int. Conf. Inf. Syst. Secur. Privacy*, 2016, pp. 407–414.

[50] K. Shim, J. Ham, B. D. Sija, and M. Kim, "Application traffic classification using payload size sequence signature," *Int. J. Netw. Manage.*, vol. 27, no. 5, p. e1981, Sep. 2017.

[51] Y. Zhai and X. Zheng, "Random forest based traffic classification method in SDN," in *Proc. Int. Conf. Cloud Comput., Big Data Blockchain (ICCBB)*, Nov. 2018, pp. 1–5.

[52] R. Bar-Yanai, M. Langberg, D. Peleg, and L. Roditty, "Realtime classification for encrypted traffic," in *Proc. 9th Int. Symp. Exp. Algorithms (SEA)*, vol. 9, Naples, Italy. Berlin, Germany: Springer, May 2010, pp. 373–385.

[53] S. Agrawal and B. S. Sohi, "Feature optimization and performance evaluation of machine learning algorithms for identification of P2P traffic," *J. Adv. Inf. Technol.*, vol. 3, no. 2, pp. 107–114, May 2012.

[54] A. Vlăduţu, D. Comăneci, and C. Dobre, "Internet traffic classification based on flows' statistical properties with machine learning," *Int. J. Netw. Manage.*, vol. 27, no. 3, p. e1929, May 2017.

[55] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[56] M. Langer, Z. He, W. Rahayu, and Y. Xue, "Distributed training of deep learning models: A taxonomic perspective," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 12, pp. 2802–2818, Dec. 2020.

[57] W. Wang, M. Zhu, J. Wang, X. Zeng, and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Jul. 2017, pp. 43–48.

[58] S. Cui, B. Jiang, Z. Cai, Z. Lu, S. Liu, and J. Liu, "A session-packets-based encrypted traffic classification using capsule neural networks," in *Proc. IEEE 21st Int. Conf. High Perform. Comput. Commun., IEEE 17th Int. Conf. Smart City; IEEE 5th Int. Conf. Data Sci. Syst. (HPCC/SmartCity/DSS)*, Aug. 2019, pp. 429–436.

[59] T. Shapira and Y. Shavitt, "FlowPic: A generic representation for encrypted traffic classification and applications identification," *IEEE Trans. Netw. Service Manage.*, vol. 18, no. 2, pp. 1218–1232, Jun. 2021.

[60] Y. Zeng, Z. Qi, W. Chen, Y. Huang, X. Zheng, and H. Qiu, "TEST: An end-to-end network traffic examination and identification framework based on spatio-temporal features extraction," 2019, *arXiv:1908.10271*.

[61] W. Wang, M. Zhu, X. Zeng, X. Ye, and Y. Sheng, "Malware traffic classification using convolutional neural network for representation learning," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2017, pp. 712–717.

[62] X. Hu, C. Gu, and F. Wei, "CLD-Net: A network combining CNN and LSTM for Internet encrypted traffic classification," *Secur. Commun. Netw.*, vol. 2021, pp. 1–15, Jun. 2021.
[63] B. Lu, N. Luktarhan, C. Ding, and W. Zhang, "ICLSTM: Encrypted traffic service identification based on inception-LSTM neural network," *Symmetry*, vol. 13, no. 6, p. 1080, Jun. 2021.
[64] W. Chen, F. Lyu, F. Wu, P. Yang, G. Xue, and M. Li, "Sequential message characterization for early classification of encrypted Internet traffic," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3746–3760, Apr. 2021.
[65] Z. Wang, P. Wang, X. Zhou, S. Li, and M. Zhang, "FLOWGAN: Unbalanced network encrypted traffic identification method based on GAN," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2019, pp. 975–983.
[66] P. Wang, S. Li, F. Ye, Z. Wang, and M. Zhang, "PacketCGAN: Exploratory study of class imbalance for encrypted traffic classification using CGAN," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2020, pp. 1–7.
[67] A. S. Iliyasu and H. Deng, "Semi-supervised encrypted traffic classification with deep convolutional generative adversarial networks," *IEEE Access*, vol. 8, pp. 118–126, 2020.
[68] S. Dong, P. Wang, and K. Abbas, "A survey on deep learning and its applications," *Comput. Sci. Rev.*, vol. 40, May 2021, Art. no. 100379.
[69] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 6999–7019, Dec. 2022.
[70] Y. Ma, F. Rusu, K. Wu, and A. Sim, "Adaptive stochastic gradient descent for deep learning on heterogeneous CPU+GPU architectures," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. Workshops (IPDPSW)*, Jun. 2021, pp. 6–15.
[71] S. Mittal, P. Rajput, and S. Subramoney, "A survey of deep learning on CPUs: Opportunities and co-optimizations," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 10, pp. 5095–5115, Oct. 2022.
[72] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, M. Hasan, B. C. Van Essen, A. A. S. Awwal, and V. K. Asari, "A state-of-the-art survey on deep learning theory and architectures," *Electronics*, vol. 8, no. 3, p. 292, Mar. 2019.
[73] Q. Zhang, L. T. Yang, Z. Chen, and P. Li, "A survey on deep learning for big data," *Inf. Fusion*, vol. 42, pp. 146–157, Jul. 2018.
[74] H. Zheng, F. Lin, X. Feng, and Y. Chen, "A hybrid deep learning model with attention-based Conv-LSTM networks for short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 11, pp. 6910–6920, Nov. 2021.
[75] G. Kang, Y. Xiao, J. Liu, Y. Cao, B. Cao, X. Zhang, and L. Ding, "Tatt-BiLSTM: Web service classification with topical attention-based BiLSTM," *Concurrency Comput., Pract. Exp.*, vol. 33, no. 16, p. e6287, Aug. 2021.
[76] S. Das, "FGAN: Federated generative adversarial networks for anomaly detection in network traffic," 2022, *arXiv:2203.11106*.
[77] C. Garbin, X. Zhu, and O. Marques, "Dropout vs. batch normalization: An empirical study of their impact to deep learning," *Multimedia Tools Appl.*, vol. 79, nos. 19–20, pp. 12777–12815, May 2020.
[78] H. Yao, C. Liu, P. Zhang, S. Wu, C. Jiang, and S. Yu, "Identification of encrypted traffic through attention mechanism based long short term memory," *IEEE Trans. Big Data*, vol. 8, no. 1, pp. 241–252, Feb. 2022.
[79] S. Soleymanpour, H. Sadr, and M. N. Soleimandarabi, "CSCNN: Cost-sensitive convolutional neural network for encrypted traffic classification," *Neural Process. Lett.*, vol. 53, no. 5, pp. 3497–3523, Oct. 2021.
[80] M. Song, J. Ran, and S. Li, "Encrypted traffic classification based on text convolution neural networks," in *Proc. IEEE 7th Int. Conf. Comput. Sci. Netw. Technol. (ICCSNT)*, Oct. 2019, pp. 432–436.
[81] M. Wang, K. Zheng, D. Luo, Y. Yang, and X. Wang, "An encrypted traffic classification framework based on convolutional neural networks and stacked autoencoders," in *Proc. IEEE 6th Int. Conf. Comput. Commun. (ICCC)*, Dec. 2020, pp. 634–641.

MEHDI SEYDALI received the B.Sc. degree in software engineering from Shahid Bahonar University, Kerman, Iran, in 2008, Iran, and the M.Sc. degree from Tarbiat Modares University, Tehran, Iran, in 2013. He is currently pursuing the Ph.D. degree in software engineering with the School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran. His research interests include big data, cloud computing, parallel processing, and distributed deep learning.

FARSHAD KHUNJUSH received the B.Sc. and M.Sc. degrees in computer engineering from Shiraz University, Shiraz, Iran, in 1991 and 1995, respectively, and the Ph.D. degree in electrical and computer engineering from the University of Victoria, Victoria, BC, Canada, in 2008. From 2008 to 2009, he was a Postdoctoral Fellow with the Laboratory for Parallel and Intelligent Systems (LAPIS). He joined the Department of Computer Science and Engineering, Shiraz University, where he is currently an Associate Professor. He was the Head of System/Software Engineering, from 2010 to 2014, and was acting as the Director of the Shiraz University ICT Center, from 2014 to 2020. He was a Visiting Professor with EPFL, Lausanne, Switzerland, in 2019, and the University of Toronto, Canada, from 2020 to 2021. He is an entrepreneur and co-founded a cloud computing startup in Iran. Since 2022, he has been an Adjunct Professor with the University of Victoria. From 2023 to 2024, he is an invited Professor with EPFL, Switzerland. His research interests include multi-core and parallel computer architectures, multi-core and parallel programming paradigms, cloud computing, and big-data processing.

BEHZAD AKBARI received the B.S., M.S., and Ph.D. degrees in computer engineering from the Sharif University of Technology, Tehran, Iran, in 1999, 2002, and 2008, respectively. He is currently an Associate Professor of computer engineering with Tarbiat Modares University. His research interests include computer networking, multimedia networking, cloud computing and networking, SDN, network virtualization and resource management in 5G networking, AI-based networking, network QoS, network performance modeling and analysis, network security, and cyber-physical systems security.

JAVAD DOGANI received the B.Sc. degree in software engineering from Technical and Vocational University, Shiraz, Iran, in 2009, and the M.Sc. and Ph.D. degrees in software engineering from the School of Electrical and Computer Engineering, Shiraz University, Shiraz, in 2012 and 2023, respectively. His research interests include distributed systems, cloud/fog/edge computing, machine learning, distributed learning, parallel computing, and big-data processing.