Received: October 28, 2024. Revised: December 18, 2024. 1200

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Adversarial Training for Improved VPN Traffic Classification Using EfficientNet-B0 and Projected Gradient Descent

Thulfiqar Mahmood Tawfeeq[1]* **Mohsen Nickray[1]**

[1]*Information Technology Department, faculty of engineering, Qom University, Qom, Iran*
* Corresponding author's Email: mahtho66@gmail.com

Abstract: This paper addresses the challenges of VPN/Non-VPN classification in increasingly complex network traffic by proposing an adversarial training-based classification method. The model integrates the EfficientNet-B0 architecture with a biLSTM structure to handle variable-length packet sequences, transforming them into fixed-size feature vectors for further extraction. Two fully connected layers increase the feature dimensions, preparing the data for convolutional processing. By incorporating Projected Gradient Descent adversarial training, the model is fine-tuned for robust classification against adversarial attacks, such as packet delays and congestion, while maintaining strong performance in normal traffic. Experimental results show high classification accuracy, achieving 99.81% on normal traffic and 99.35% on adversarial traffic using the ISCX 2016 dataset, with minimal trade-offs. This approach offers a scalable and resilient solution to VPN/Non-VPN classification in hostile network environments.

Keywords: Adversarial training, BiLSTM network, EfficientNet-B0, Projected gradient descent, VPN traffic classification.

1. Introduction

The rapid expansion of the internet and the increased use of Virtual Private Networks (VPNs) have led to a growing need for effective and accurate network traffic classification. Differentiating between VPN and Non-VPN traffic is critical for tasks such as traffic management, security monitoring, and anomaly detection. However, classifying encrypted VPN traffic presents unique challenges, as encryption conceals packet contents, making traditional feature extraction and classification methods ineffective. Many existing approaches rely heavily on manual feature engineering or utilize pre-processed and clipped data to simplify input sequences. These methods can lead to information loss and are often unable to adapt to the complexities of real-world network traffic, where dynamic, variable-length packet data must be processed efficiently and with high accuracy.

Further complicating the task, modern network environments are prone to adversarial conditions, where even small disruptions, such as packet loss, network congestion, or intentional adversarial attacks, can affect classification performance. Models trained solely on normal traffic data often fail to generalize well in these adverse scenarios, leading to misclassifications and decreased reliability. This necessitates the development of robust classification models that can handle encrypted VPN traffic and maintain high accuracy in both normal and adversarial environments. Thus, a solution that integrates advanced deep learning techniques with adversarial robustness is required to address these challenges and improve the overall effectiveness of network traffic classification. Numerous methods have been proposed to improve network traffic classification performance under various conditions.

In [1], a comprehensive review of network traffic classification techniques is presented, covering methods such as port-based techniques, deep packet inspection, and the use of statistical features combined with machine learning and deep learning algorithms. The challenges and future opportunities in this domain are also discussed. In [2], a novel method is proposed using an unsupervised deep

learning model consisting of a Convolutional Neural Network (CNN) and an Autoencoder for the early detection of anomalous traffic. This approach achieves high accuracy. However, the proposed D-PACK model in this paper only examines the first few packets in each flow and cannot examine all packets. This can cause challenges if the malicious pattern is hidden in the unseen packets.

A novel approach using deep Convolutional Recurrent Neural Networks (CRNN) for extracting spatio-temporal features is introduced in [3]. The paper demonstrates that by capturing more complex features, significant improvements in classification accuracy are achieved over traditional methods. A limitation of this method is its dependency on predefined statistical features like packet counts and inter-arrival times, which may overlook finer-grained packet-level details. Reference [4] introduces an improved Support Vector Machine (SVM) algorithm, CMSVM, to address data imbalance issues in network traffic classification. The use of active learning in this method enhances both accuracy and performance. Nevertheless, incorporating additional features into the dataset or applying the CMSVM approach to other machine learning algorithms could potentially lead to better performance according to the authors' declaration. A multi-scale feature attention approach using CNNs is presented in [5] for network traffic classification. This method demonstrates higher accuracy than existing techniques by analyzing the initial packets of network flows. It takes only one packet per flow for network traffic classification. In [6], a new platform is introduced for encrypted traffic classification, using statistical features of network flows and employing machine learning algorithms such as Extreme Gradient Boosting (XGBoost) and LightGBM, achieving promising results. A comprehensive review of network traffic classification techniques is provided in [7], categorizing methods into five main types: statistical classification, correlation-based, behavior-based, content-based, and port-based. The paper also proposes evaluation criteria for these methods. In [8], the application of deep learning for traffic classification is explored, with an emphasis on identifying encrypted traffic and addressing the limitations of traditional port-based and content-based methods. In [9], a review of data mining and machine learning methods for traffic classification in sustainable smart cities is presented. The paper discusses the challenges associated with data complexity and feature selection in traffic classification. A hybrid neural network method, Deep Multiscale Hybrid Neural Network (DM-HNN), for analyzing both flow-level and packet-level features is introduced in [10]. This dual-feature extraction approach shows improved performance in traffic classification over single-mode models. Reference [11] presents a novel method for classifying network traffic based on timing features extracted from network packets. This approach distinguishes between encrypted and unencrypted traffic with high accuracy, promising for real-time traffic analysis. Although the method can classify VPN traffic quickly, it achieves 95.02% accuracy which needs more enhancement. Moreover, all aforementioned methods do not consider adversarial data and artifacts that are prevalent in network traffic data.

A multi-task learning method called "Multi-task Transformer (MTT)" is proposed in [12]. This model simultaneously performs application identification and traffic classification, outperforming other methods and achieving faster results. Unlike the other methods, this paper considers the packets as the sequence of bytes for input. However, it can't handle variable length packet size in the flows. In [13], a deep learning approach called "Deep Packet" is introduced, combining a CNN and an Autoencoder to classify VPN and non-VPN traffic, yielding better results than existing methods in the identification of encrypted traffic. This method also takes a fixed size of packet input equal to 1480 bytes. Reference [14] explores traffic classification methods in software-defined networks (SDN) using machine learning models such as SVM, nearest centroid, and Naïve Bayes, achieving good classification accuracy. The proposed method in [14], however, uses some statistical attributes from input packet flows. In [15], various machine learning techniques, including decision trees, random forests, and recurrent neural networks, are applied to network traffic classification. The paper highlights the potential of these techniques in addressing real-world classification challenges. A novel method for network traffic classification using federated semi-supervised learning is presented in [16]. This approach demonstrates high performance while maintaining minimal accuracy gaps compared to centralized training. However, the adversarial data are not considered in [15, 16]. In [17], a semi-supervised approach based on deep learning for network traffic classification is introduced. This method addresses the challenge of utilizing both labeled and unlabeled data, which is often overlooked in supervised learning approaches. While the effect of noise and artifacts is reduced using denoising autoencoder in [17], the raw packages are not incorporated in this method. Instead, some statistical features are used as input. Reference [18] proposes the SSDDQN model, where the current network uses an autoencoder to reconstruct traffic features and a

deep neural network for classification. The target network employs K-Means clustering for unsupervised learning, followed by deep neural network prediction. However, this approach uses well-known datasets such as NSL-KDD with statistical predefined inputs. A novel approach for classifying VPN and non-VPN traffic, as well as identifying encrypted applications, is presented in [19]. The inclusion of entropy as an additional feature enhances performance, demonstrating high accuracy across all three machine learning algorithms. However, it doesn't consider adversarial data and noises. Reference [20] proposes the use of multiple Multilayer Perceptron (MLP) convolutional layers in the NIN model to map fixed-length packet vectors to application or traffic labels. Additionally, a parallel decision strategy is employed to process packet headers and packet bodies separately, improving classification accuracy. This paper uses raw packets as input to ensure most patterns are kept, but it can't handle variable-length input since its input packet sizes are fixed after padding or truncating. In [21], a network encryption traffic classification model combining attention mechanisms and spatiotemporal features is introduced. This method focuses on addressing the challenges of classifying both encrypted and unencrypted network traffic, particularly the underlying traffic of encrypted applications. The proposed method in [21] also works with raw traffic packets. However, since it makes all inputs have a uniform size of 784 bytes, it might lose some important information and patterns. A novel approach for efficient and accurate network traffic classification using ensemble machine learning models is presented in [22]. This method demonstrates superior performance in both multiclass and binary classification, particularly excelling with Light GBM when employing min-max scaling. The method uses hand-crafted statistical features from eight properties of the network flow. Reference [23] outlines a two-step process: first, preprocessing five-tuple data and analyzing probabilistic feature values using first-order Markov chains, which serve as input for the CNN model. An improved adaptive step method optimizes the CNN training, and the bagging algorithm is integrated to enhance classification performance. The method achieves a maximum of around 98% accuracy which can be improved further. In [24] a Domain Name System-based (DNS) security approach is proposed to detect incidents in network data by analyzing entropy changes in DNS query traffic. This method investigates variations in entropy values based on unique source IP addresses and DNS query keywords from internal and external network traffic, revealing distinct entropy patterns associated with recent spam bot activity. The analysis concludes that abnormal entropy changes, especially parallel and symmetrical types, indicate spam bot attempts to exploit local email servers for spreading spam. A novel approach for network traffic classification using A-DBSCAN and standard classifiers is presented in [25]. The experiments conducted on the ISCX VPN and Non-VPN dataset demonstrate that the proposed model achieves an accuracy of 81.9%. In [26], network traffic classification is improved using the AutoEncoder-Support Vector Machine with Gaussian Optimization (AE-SVM-GO) model to tackle data imbalance and overfitting in intrusion detection. This hybrid method combines an autoencoder for balancing classes, SVM for classification, and Grasshopper Optimization for hyperparameter tuning, achieving 95.3% accuracy.

Despite the advancements in network traffic classification, existing studies may face significant challenges, particularly in handling adversarial attacks and variable-length packet data. Many traditional methods rely on manual feature extraction, which can limit the model's ability to generalize to new and unseen traffic patterns. Additionally, conventional models may struggle to maintain high classification accuracy when faced with adversarial data, leaving critical gaps in ensuring robust performance under real-world network conditions. These limitations highlight the need for more sophisticated architectures that can adapt to the dynamic nature of network traffic and provide resilience against adversarial threats.

To address these issues, this paper proposes a novel method that integrates the EfficientNet-B0 architecture with projected gradient descent adversarial training. The primary contribution of this paper is the development of a novel network traffic classification approach combining the powerful pre-trained EfficientNet-B0 structure with Bidirectional Long Short-Term Memory (BiLSTM) for adaptive feature extraction from variable-length traffic packet sequences. Unlike previous methods that rely on normal training, the proposed model not only handles variable-length traffic packet sequences but also enhances robustness against adversarial attacks. By leveraging transfer learning and fine-tuning a pre-trained EfficientNet-B0, the model effectively extracts features from raw traffic data without the need for manual preprocessing. The addition of adversarial training further strengthens the model's resilience, enabling it to accurately classify both normal and adversarial traffic with high precision.

The organization of this paper is as follows: Section 2 introduces the foundational concepts necessary for understanding the proposed method.

Section 3 outlines the methodology, including the design of the network and the implementation of Projected Gradient Descent (PGD) adversarial training. Sections 4 and 5 describe the dataset and the evaluation metrics used to assess the performance of the proposed approach. Section 6 presents the simulation results, covering performance evaluations across different lambda values, as well as the classification outcomes for both normal and adversarial network traffic. A comparison between the proposed method and existing approaches in the literature is provided in Section 7. Finally, Section 8 concludes with a discussion of the implications of the findings.

2. Basic concepts

This section provides a comprehensive overview of the essential principles and core concepts necessary for understanding the proposed method

2.1 The structure of pre-trained EfficientNet-B0

EfficientNet-B0 is widely available as a pre-trained model, meaning it has already been trained on large datasets such as ImageNet, which contains millions of labeled images across thousands of categories. Using a pre-trained model offers significant advantages, as it allows the model to leverage previously learned features, accelerating convergence and improving performance on new tasks, especially when data is limited. By fine-tuning a pre-trained EfficientNet-B0, users can adapt the model to their specific task without starting from scratch. EfficientNet-B0 follows a structured design, organized into several layers and blocks, each optimized for efficient feature extraction while maintaining a small parameter footprint [27]. The key components of its structure include:

Initial Convolution Layer: The first layer of EfficientNet-B0 consists of a 3×3 convolution with 32 filters, applied to the input image. This layer operates with a stride of 2, reducing the spatial resolution of the image while extracting initial features. This is followed by batch normalization and Swish activation to stabilize training and improve non-linear feature extraction.

Mobile Inverted Bottleneck Convolution (MBConv) Blocks: The core of EfficientNet-B0 is built around the MBConv blocks, which are an efficient variant of the traditional bottleneck block. The structure of this module is illustrated in Fig. 1. Each MBConv block contains three stages:

A 1×1 expansion convolution to increase the number of channels and enable richer feature extraction.

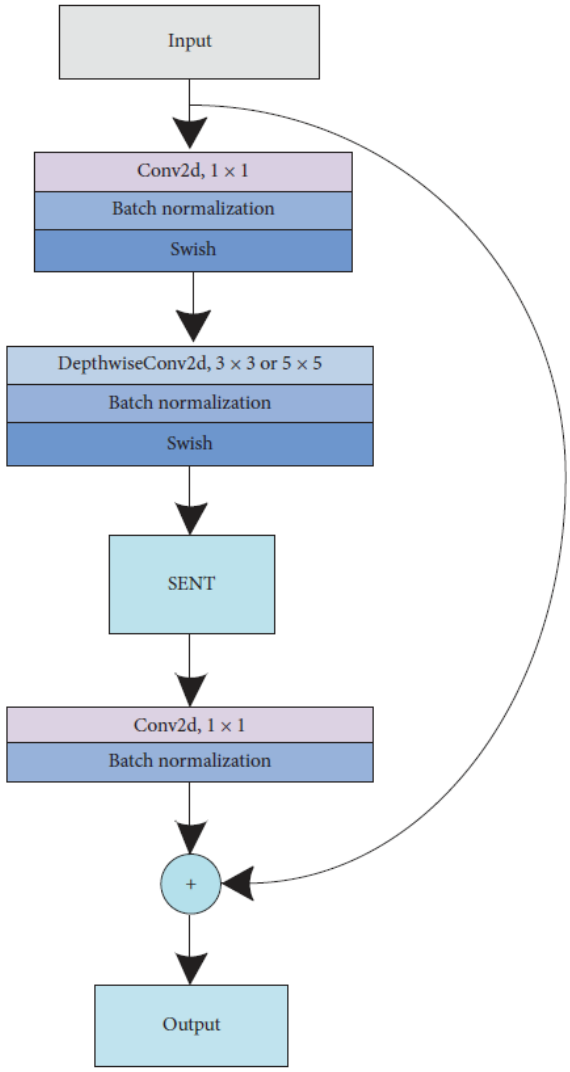

Figure. 1 Depiction of the MBConv framework [29]

A depthwise convolution (either 3×3 or 5×5) to efficiently extract spatial information.

A 1×1 projection convolution that reduces the number of channels back to the original size, making the block computationally efficient.

Additionally, a residual connection is applied when the input and output dimensions match, which helps retain information from earlier layers.

Squeeze-and-Excitation (SE) Block: After the depthwise convolution in each MBConv block, EfficientNet-B0 uses an SE block to recalibrate the feature maps. The SE block compresses the feature map using global average pooling, applies a sigmoid activation to calculate attention weights for each channel, and multiplies these weights by the original feature map. This mechanism enhances the most important channels, improving the model's focus on relevant features [28].

Swish Activation: EfficientNet-B0 uses the Swish activation function, which is defined as the Eq. (1).

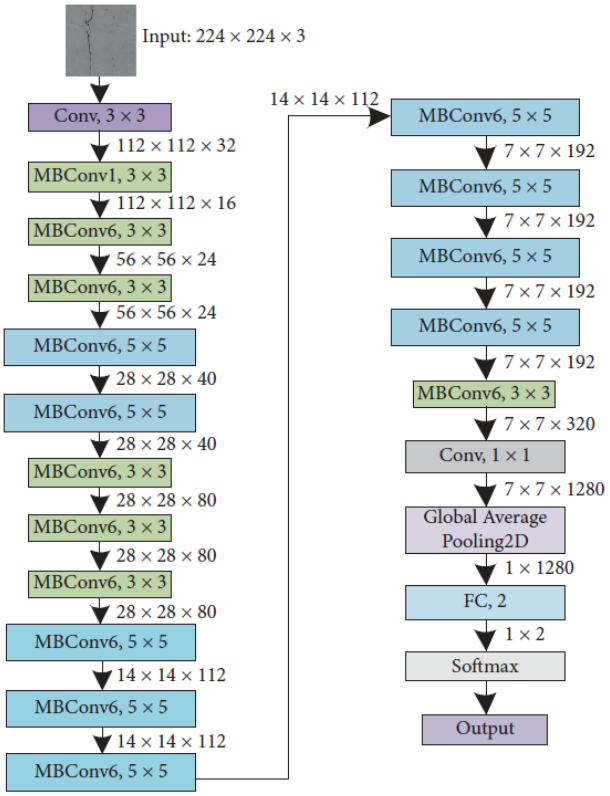

Figure. 2 The architecture of the EfficientNetB0 network [29]

$$Swish\ (x) = x.sigmoid\ (x) \tag{1}$$

Swish has been shown to outperform traditional activation functions like Rectified Linear Unit (ReLU) by enabling smoother gradients and better optimization during training.

Global Average Pooling: After passing through several MBConv blocks, EfficientNet-B0 uses a global average pooling layer, which condenses the entire spatial feature map into a single vector per channel. This process reduces the number of parameters while preserving global information, making the model more efficient.

Fully Connected Layer: The global pooled features are fed into a fully connected layer with 1000 output neurons, corresponding to the number of classes in the ImageNet dataset. In fine-tuning tasks, this final layer can be modified to match the number of classes in the target task.

The overall architecture of EfficientNetB0 is shown in Fig. 2.

2.2 Projected gradient descent adversarial training

PGD is a widely used method in adversarial training that aims to improve the robustness of machine learning models against adversarial attacks. PGD works by perturbing a clean example x0x_0x0 iteratively, gradually moving towards a more adversarial version of the input while maintaining the perturbation within a predefined constraint, typically an ϵ\epsilonϵ-ball around the original example. This ensures that the adversarial example remains imperceptibly different from the original input.

In PGD, the adversarial example is generated over K iterative steps, with each step applying a small perturbation to the input. The perturbation at each step is guided by the gradient of the loss function with respect to the input, pushing the input in the direction that maximizes the model's loss. The update for the adversarial example at step k is given by Eq. (2).

$$X_k = \prod(X_{k-1} + \alpha.sign(\nabla_x \mathcal{L}(f_\theta(X_{k-1}), y))) \tag{2}$$

The projection step ensures that the adversarial example remains within the permissible perturbation range, maintaining its similarity to the original input. By iterating over multiple steps, PGD effectively produces adversarial examples that expose the vulnerabilities of the model [30].

To train models with PGD, adversarial examples generated using the PGD algorithm are included in the training process. The model is trained not only on clean examples but also on perturbed adversarial examples. This approach forces the model to learn more robust decision boundaries that are less susceptible to small, adversarial perturbations. PGD adversarial training is known for offering high levels of robustness against a variety of adversarial attacks, as demonstrated in [31].

3. Methodology

In this section, the methodology used to develop the proposed network traffic classification system is described in detail. The goal of this method is to classify network traffic into two categories: VPN traffic and Non-VPN traffic. To ensure that no important information is lost, the raw packet data from the network is used as the input, allowing for the automatic extraction of features specifically designed for traffic classification. Unlike previous approaches that relied on manually extracted features [32] or preprocessed data through clipping or padding [33], this method directly processes raw network packets. These packets often consist of complex, variable-length sequences that are challenging to handle. One of the key contributions of this work is the introduction of a novel network architecture that can process raw packet data without the need for additional preprocessing steps such as clipping,

padding, or converting the data into images. In designing this architecture, the first step was to choose a base deep learning structure (i.e. convolutional-based structure, recurrent-based structure, etc.). For this work, a convolutional-based structure was selected as the core of the model. Convolutional layers are well-suited for detecting patterns regardless of their position in the input data due to their translation invariance, a key advantage when analyzing network traffic where specific patterns can appear at various positions within packet sequences. Additionally, their ability to capture spatial dependencies and build hierarchical representations of the input data makes them effective for processing network traffic data. These layers are also highly computationally efficient, a crucial factor for our purpose, as it allows for the design of a large and complex network capable of processing large-scale network traffic data without significantly increasing the training costs. Rather than building and training a CNN from scratch, we leverage transfer learning by fine-tuning a pre-trained network to further enhance computational efficiency. Many state-of-the-art pre-trained CNN architectures are designed and trained on large-scale, complex tasks, making them well-suited for transfer learning with only a few epochs of fine-tuning for new tasks. Among these architectures, EfficientNet-B0 is selected for this work.

EfficientNet-B0 belongs to the EfficientNet family, which is designed to deliver high performance with minimal computational cost. Unlike other models that scale in just one direction (e.g., increasing depth or width), EfficientNet-B0 balances scaling across depth, width, and resolution to optimize for both accuracy and efficiency. This balanced scaling gives EfficientNet-B0 a superior accuracy-to-FLOPS ratio compared to older architectures like Residual Network (ResNet) and Inception, making it one of the most computationally efficient models. However, before fine-tuning it for our task, several modifications are required to adapt the network to our problem.

One critical modification involves adjusting the input layer. While EfficientNet-B0 expects inputs of 224x224 spatial dimensions with 3 channels, the raw traffic packet data consists of single-channel sequences with variable temporal lengths. To address this, a novel structure is introduced at the beginning of EfficientNet-B0 to extract features from traffic packets before feeding them into the network. This structure consists of two bidirectional LSTM layers, each with 240 hidden units, chosen based on the average packet size in the traffic data. While some packets can exceed 3000 temporal dimensions, the average packet size is typically below 400. Since BiLSTM layers process data in both directions, the output of the first BiLSTM layer is a tensor with 480 channels, maintaining the original temporal dimension.

The second BiLSTM is configured to output only the final temporal state, resulting in a tensor with 480 channels and no temporal dimension. This effectively transforms the variable-length sequences into fixed-size tensors with 480 channels. In other words, this structure automatically extracts features from variable-length input sequences and projects them into a fixed-size tensor, allowing the EfficientNet-B0 to process the data. However, this feature extraction process alone is insufficient for use in a deep architecture like EfficientNet-B0.

To further refine the feature extraction process and expand the data's dimensionality, two fully connected layers are added immediately after the BiLSTM layers. The first fully connected layer consists of 1000 neurons, chosen to maintain a hierarchical feature extraction process rather than making an abrupt jump from 480 initial features to a much larger number. The second fully connected layer contains 9408 neurons. The decision to use 9408 neurons in this layer is linked to the next set of layers in the proposed architecture.

As mentioned earlier, EfficientNet-B0 is originally designed to process images with dimensions of 224x224x3, which translates to 150,528 pixels. However, extracting 150,528 features directly would require considerable computational resources. Instead, we first extract 9408 features—approximately 1/16 of the original pixel count—using fully connected layers. The data is then projected and reshaped into a format of 56x56x3, where 56x56 represents the spatial dimensions and 3 channels are maintained. To match the required input dimensions of EfficientNet-B0, a resizing layer is employed, with a scaling factor of [4,4], which resizes the data to 224x224x3.

This architectural design ensures that the key features are extracted from the variable-length sequences automatically, without manual intervention. The process is fully optimized and learned during the network's training phase. In the final step, the last fully connected and SoftMax layers of EfficientNet-B0, which are originally designed for image classification, are replaced with two new layers suited for binary classification, specifically for distinguishing between VPN and Non-VPN network traffic.

Once the network is designed, it could be fine-tuned using a standard training procedure. However, normal training might leave the network vulnerable

to various natural network phenomena that are not represented in the training dataset. For instance, delays or packet drops due to congestion could produce patterns not present in the training data, and legitimate shifts in routing or connection handovers (such as switching between Wi-Fi and mobile networks) could introduce variations in traffic flow. These types of changes could be misinterpreted as adversarial data. Adversarial data refers to input that has been intentionally manipulated to mislead a machine learning model into making incorrect predictions or classifications.

To ensure the proposed network is robust against such adversarial data, adversarial training is used instead of the normal training procedure as a key enhancement. Specifically, this paper employs the projected gradient descent method to replace the traditional training approach. PGD is an iterative adversarial attack technique that generates adversarial examples by applying small perturbations to the input data. These perturbations maximize the model's loss while ensuring that the perturbed data remains within a defined constraint. PGD is particularly effective because it generates stronger and more reliable adversarial examples compared to other methods, such as the fast gradient sign method (FGSM). By incorporating PGD-based adversarial training, the model becomes more robust to both simple and complex variations in network traffic flow, reducing the risk of incorrect classifications due to unseen or adversarial inputs.

Let θ be the network initial parameters of the network (the pre-trained weights for the EfficientNet-B0 part and random values for the new parts) and $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$ be the training data with input samples x_i, corresponding labels y_i, and total number of samples *N*. In each epoch of training, the training data will be shuffled, and a mini-batch of data $B = \{(x_1, y_1), (x_2, y_2), \dots, (x_b, y_b)\}$ with the size of *b* equal to 40 is selected. Next, for each input *x* in the mini-batch adversarial examples are generated using PGD. This consists of several iterative steps starting with initializing perturbation. In this regard, the initial perturbation δ_0 is calculated as small random noise bounded by ϵ as follows:

$$\delta_0 = \mathcal{U}(-\varepsilon, \varepsilon) \tag{3}$$

ϵ normally ranges from 2 to 8 for 8-bit images (pixels range from 0 to 255). Hence, ϵ is considered equal to 0.015 (between 2/255 to 8/255) for the normalized traffic packets (normalized data range from 0 to 1) in this paper.

Next, the perturbation value is obtained in an iterative process (iterative gradient ascent). In this regard, for each iteration *k* of PGD (from *k*=0 to *K*−1 where *K*=3 is the number of iterations for generating adversarial examples) the gradient of the loss function $L(f_\theta(x + \delta_k), y)$ is calculated with respect to the perturbed input $x + \delta_k$ as follows:

$$g_k = \nabla_x L(f_\theta(x + \delta_k), y) \tag{4}$$

Next, the perturbation δ_k is updated by taking a step in the direction of the gradient using Eq. (5).

$$\delta_{k+1} = \delta_k + \alpha \cdot sign(g_k) \tag{5}$$

In this paper, α is considered 0.005 (ϵ / *K*). Finally, δ_{k+1} is clipped to ensure that the perturbation stays within the allowed bound ϵ:

$$\delta_{k+1} = max(-\varepsilon, min(\delta_{k+1}, \varepsilon)) \tag{6}$$

After *K* iterations, the final adversarial example is $x' = x + \delta_k$, where δ_K is the perturbation found at the last PGD iteration.

Once the adversarial examples are generated, the model is trained on both normal and perturbed inputs. In this regard, both adversarial loss L_{adv} and normal loss L_{normal} are calculated as Eqs. (7) and (8), respectively.

$$L_{adv} = L(f_\theta(x'), y) \tag{7}$$

$$L_{normal} = L(f_\theta(x), y) \tag{8}$$

The total loss is finally calculated using both adversarial and normal losses and the adversarial rate parameter λ which makes a trade-off between the importance of normal examples and adversarial ones for training the network. The Eq. (9) shows how it is calculated as mentioned.

$$L_{Tot} = \lambda L_{adv} + (1 - \lambda) L_{normal} \tag{9}$$

After computing the total loss, the model parameters θ are updated using gradient descent. In this regard, the gradient of the total loss is calculated with respect to the model parameters θ using Eq. (10).

$$g_\theta = \nabla_\theta L_{adv} \tag{10}$$

Finally, the gradient descent update is performed to minimize the total loss using the Adaptive Moment Estimation (ADAM) optimization algorithm. After updating the model parameters, the process continues for each mini-batch in the training dataset. The overall process is also repeated for 30 epochs until the

model converges. The reason behind choosing the small value of 30 for the number of epochs is the ability of the proposed method to learn fast due to the network structure, transfer learning, and customized PGD learning algorithm. By adopting 30 epochs for training the network, this ability can be evaluated.

4. Dataset

The ISCX 2016 VPN and Non-VPN Traffic dataset is employed in this paper to evaluate the proposed method. The ISCX 2016 was created to provide a comprehensive and representative collection of real-world network traffic [34]. The primary objective in constructing this dataset was to ensure a sufficient diversity and quantity of traffic types, simulating real-world scenarios for research purposes. To achieve this, the dataset creators defined a set of tasks, ensuring that the traffic captured represented common online activities. User accounts for Alice and Bob were created to use popular services such as Skype, Facebook, and others, ensuring the dataset reflected typical online behavior. The dataset comprises traffic from both regular network sessions and sessions routed over a VPN. In total, 14 traffic categories were generated, including VoIP, VPN-VoIP, P2P, and VPN-P2P, among others. For further clarification of the dataset composition, the distribution of traffic types based on percentage is as follows: Browsing accounts for 15% of the total traffic, Email makes up 10%, Chat constitutes 8%, Streaming represents 20%, File Transfer is 12%, VoIP contributes 10%, P2P is 15%, and VPN Traffic accounts for 10%. This distribution provides a comprehensive representation of real-world network activities and covers various types of traffic, such as browsing, streaming, and communication applications, both with and without VPN. However, for this work, we will only use the data related to VPN traffic. Table 1 provides a detailed description of the different types of traffic and applications captured:

The traffic was captured using network packet analyzers such as Wireshark and tcpdump, resulting in a total dataset size of 28 GB. VPN traffic was generated using an external VPN provider connected via OpenVPN (UDP mode). To generate SFTP and FTPS traffic, an external service provider was used in conjunction with the FileZilla client. To streamline the labeling process, all non-essential services and applications were disabled during data capture, ensuring that only relevant traffic was recorded (e.g., Skype voice calls, SFTP file transfers). Filters were applied to capture only packets with source or destination IP addresses associated with the local client. The dataset includes labeled network traffic in two formats: full packet captures (pcap) and CSV

Table 1. description of the different types of traffic

Traffic	Description
Browsing	This category includes HTTPS traffic generated by users during browsing sessions or tasks involving a web browser. For instance, while capturing voice calls using Google Hangouts, browsing traffic was also recorded, even though browsing was not the primary activity.
Email	Email traffic was captured using the Thunderbird email client and Gmail accounts for Alice and Bob. The clients were configured to send emails via SMTPS and receive emails using both POP3/SSL and IMAP/SSL protocols.
Chat	This category includes instant messaging applications such as Facebook and Hangouts via web browsers, as well as Skype, AIM, and ICQ, using the Pidgin application.
Streaming	The streaming category consists of multimedia applications requiring continuous data transfer. Traffic from YouTube (HTML5 and Flash versions) and Vimeo was captured using Chrome and Firefox browsers.
File Transfer	This category includes traffic from applications designed for sending and receiving files. The dataset includes traffic from Skype file transfers, FTP over SSH, and FTP over SSL sessions.
VoIP	Voice over IP traffic was generated by applications such as Facebook, Hangouts, and Skype voice calls.
P2P	Peer-to-peer traffic was captured by downloading various .torrent files using the uTorrent and Transmission applications.

Table2. The mapping relationship between application types and traffic types

Traffic Type	Application Type
Web Browsing	Firefox and Chrome
Email	SMTPS, POP3S, IMAPS
Chat	ICQ, AIM, Skype, Facebook, Hangouts
Streaming	Vimeo, YouTube
File Transfer	Skype, FTPS, SFTP (using FileZilla and external services)
VoIP	Facebook, Skype, Hangouts voice calls (1-hour duration)
P2P	uTorrent, Transmission

files generated using ISCXFlowMeter, a Java-based tool that reads pcap files and extracts selected features. The dataset, along with the tools, is publicly available for researchers. The UNB ISCX Network Traffic Dataset includes traffic from a range of applications such as YouTube, Gmail, Facebook, Skype, and Bittorrent, ensuring that researchers can analyze various types of network behavior (Table2). This dataset is particularly valuable for developing machine learning models for traffic classification and anomaly detection, as well as studying VPN and Non-VPN traffic patterns in security research.

In this paper, the traffic packets are normalized between 0 and 1 by dividing the packet values to 255 first. Then, the dataset is balanced by undersampling.

5. Evaluation metrics

In this study, the performance of the proposed model is evaluated using four key metrics: accuracy, precision, recall, and F1 score. These metrics are widely recognized in classification tasks, as they provide a comprehensive assessment of a model's ability to classify network traffic accurately and handle imbalanced data distributions.

Accuracy: Accuracy is a fundamental metric that measures the overall correctness of the classifier by determining the ratio of correctly classified instances to the total number of instances. In the context of network traffic classification, accuracy can be formally defined as Eq. (11).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{11}$$

where:

- TP represents the number of correctly classified network flows that belong to a specific application.
- TN denotes the number of correctly classified flows that do not belong to a specific application.
- FP refers to the number of flows incorrectly classified as belonging to an application.
- FN refers to the number of flows belonging to an application but classified otherwise.

While accuracy is useful in scenarios with balanced classes, it may not provide a complete understanding of the classifier's performance in cases of class imbalance, which is common in network traffic classification.

Precision: Precision measures the proportion of correctly predicted instances out of all instances predicted to belong to a specific class. It is particularly important in network traffic classification for understanding the classifier's ability to minimize false positives, especially when certain traffic types are rare or critical. Precision is computed as Eq. (12).

$$Precision = \frac{TP}{TP+FP} \tag{12}$$

A high precision score indicates that the model effectively identifies true traffic instances with minimal misclassification, which is essential when distinguishing between different network applications.

Recall: Recall, also known as sensitivity or true positive rate, measures the proportion of actual instances of a class that are correctly classified. It assesses the model's ability to detect relevant network flows, particularly minimizing false negatives. Recall is defined as Eq. (13).

$$Recall = \frac{TP}{TP+FN} \tag{13}$$

In network traffic classification, high recall is essential when the goal is to ensure that important or security-sensitive traffic types are not overlooked by the model.

F1 Score: The F1 score is the harmonic mean of precision and recall, offering a balanced measure that accounts for both false positives and false negatives. It is particularly useful in network traffic classification when the dataset is imbalanced, and optimizing both precision and recall is crucial. The F1 score is computed as Eq. (14).

$$F_1 Score = \frac{2\times(Precision\times Recall)}{Precision+Recall} \tag{14}$$

The F1 score provides a single metric to evaluate the trade-off between precision and recall, making it a valuable tool for assessing the performance of the classification model across different types of network traffic.

6. Simulation results

This section presents the simulation results for the proposed network traffic classification model and evaluates its performance across various scenarios. The analysis assesses the effectiveness of the EfficientNet-B0-based architecture and examines how PGD adversarial training influences detection accuracy. Key evaluations include lambda parameter adjustment, performance metrics for normal traffic

classification, and the model's resilience against adversarial traffic. The simulations were performed using MATLAB 2024a on a system with an Intel Core i7 13650HX CPU, 16 GB RAM, and an NVIDIA RTX 4060 GPU featuring 8GB of GDDR6 memory.

6.1 Lambda parameter adjustment

In this section, the impact of the λ parameter on the performance of the proposed network traffic classification model, which incorporates PGD adversarial training, is evaluated. The parameter λ controls the balance between normal and adversarial data during the training process. Its effect on classification accuracy is tested across three datasets, namely training, testing, and adversarial data. Four values of λ, including 0 (representing normal training), 0.25, 0.5, and 0.75 are explored. The results, illustrated in Fig. 3, highlight how performance varies with different values of λ across the datasets.

Starting with the training data, the accuracy remains consistently high across all λ values. With λ set to 0 (i.e., normal training), the accuracy reaches 99.8843%. When adversarial training is introduced with λ = 0.25, the accuracy shows a slight increase to 99.9074%. As λ continues to increase, we observe minor drops in performance: 99.8611% for λ = 0.5, and 99.6529% for λ = 0.75. These results suggest that while adversarial training provides enhanced robustness, placing too much emphasis on adversarial data slightly reduces accuracy on normal training data.

Turning to the testing data, the model trained with λ = 0 achieves an accuracy of 99.7222%. Interestingly, when λ is set to 0.5, the model reaches its highest testing accuracy at 99.8148%, surpassing the performance of both normal training and other λ values. At λ = 0.25, the accuracy is 99.4444%, and with λ = 0.75, it decreases further to 99.3519%. This suggests that moderate λ values, particularly 0.5, help the model generalize better in testing scenarios, as adversarial training forces the network to adapt to more challenging conditions, expanding its decision-making boundaries.

The most striking differences emerge in the model's performance under adversarial attacks. Unsurprisingly, the model trained without adversarial data (λ = 0) struggles significantly in this scenario, achieving only 92.7778% accuracy. This is expected, as the network has not been exposed to adversarial data during training. However, when adversarial training is introduced with λ = 0.25, the model's accuracy rises sharply to 98.7037 %.

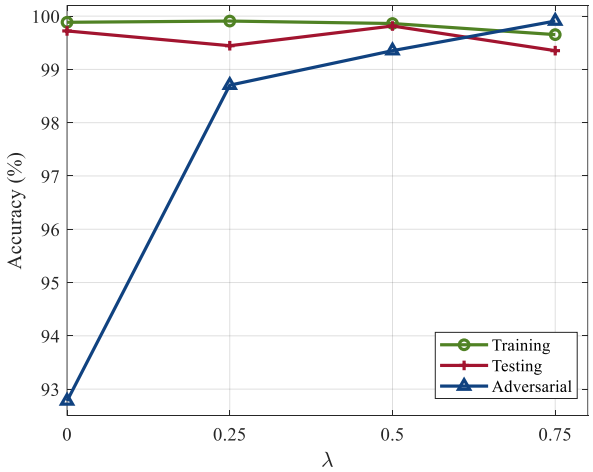

Figure. 3 Evaluation of the performance for different λ values

The trend continues with λ = 0.5 yielding 99.3519%, and λ = 0.75 achieving the highest accuracy of 99.9074 %. These results demonstrate the significant boost in resilience provided by PGD adversarial training, particularly when higher λ values are used.

All in all, the results indicate that a moderate λ value of 0.5 provides an optimal balance between performance on normal and adversarial data. While higher λ values, such as 0.75, offer stronger protection against adversarial attacks, they do so at the cost of a slight decline in accuracy on normal data. Based on these findings, we select λ = 0.5 for the remaining experiments, as it offers the best trade-off between robustness and generalization.

6.2 Normal traffic classification results

This section presents the performance of the proposed method in classifying normal network traffic, using both a confusion matrix and a Receiver Operating Characteristic curve for evaluation.

A confusion matrix is an effective tool for assessing the performance of a classification model by comparing predicted labels with true labels. It provides a breakdown of the model's TP, TN, FP, and FN, offering insights into the model's accuracy, precision, recall, and overall reliability. The diagonal elements of the confusion matrix represent correct predictions, while the off-diagonal elements indicate misclassifications. In this context, TP correspond to non-VPN instances correctly identified, and TN represent VPN instances that were correctly classified. FP occur when VPN instances are misclassified as non-VPN, while FN refer to non-VPN instances that were incorrectly predicted as VPN.

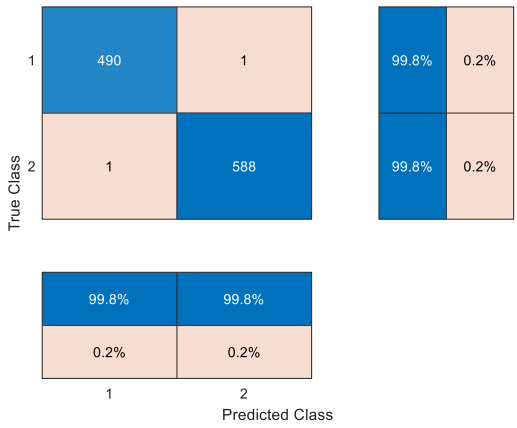

Figure. 4 The confusion matrix of evaluating the testing dataset using the proposed approach

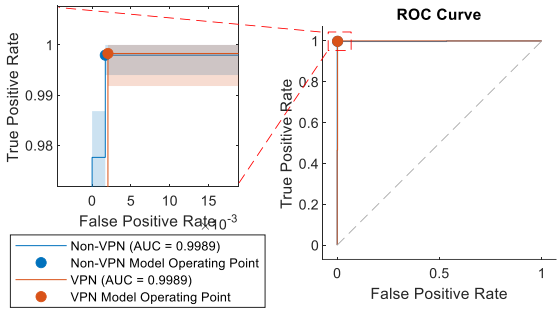

Figure. 5 The ROC curve of evaluating the testing dataset using the proposed approach

Fig. 4 illustrates the confusion matrix for the proposed model's performance on normal traffic classification. The results show that the model accurately classified 588 VPN samples (TNs) and 490 Non-VPN samples (TPs). There was only 1 FP and 1 FN, underscoring the model's high precision and recall. These results highlight the model's exceptional ability to distinguish between VPN and Non-VPN traffic packets, with minimal misclassifications. The low number of false positives and false negatives further demonstrates the model's reliability in classifying VPN traffic while maintaining a very low error rate for Non-VPN samples.

The Receiver Operating Characteristic curve (ROC) is another crucial metric used to evaluate classification performance. The ROC curve plots the true positive rate (sensitivity) against the false positive rate (1-specificity) at various decision thresholds. The Area Under the Curve (AUC) represents the model's overall ability to differentiate between classes. An AUC close to 1 indicates that the model is highly effective at distinguishing between positive and negative classes, while an AUC close to 0.5 suggests poor performance. The ROC curve is especially useful for imbalanced datasets, where one class is more prevalent than the other.

Fig. 5 presents the ROC curve for the proposed model's classification of normal network traffic. The AUC for this curve is an impressive 0.9999, which is nearly the maximum possible value, reflecting the model's near-perfect classification performance. This extremely high AUC underscores the model's accuracy in distinguishing between VPN and non-VPN traffic, while minimizing false positives and false negatives. The ROC curve's proximity to the top-left corner of the plot further confirms the model's robustness and effectiveness, achieving a high true positive rate while keeping the false positive rate very low in normal traffic scenarios.

6.3 Adversarial traffic classification results

This section analyzes the performance of the proposed model when classifying adversarial network traffic and compares it with the results for normal data to assess the model's robustness. Fig. 6 shows the confusion matrix for adversarial traffic classification. The model demonstrates strong performance, correctly identifying 487 TPs and 586 TNs, while only misclassifying 3 instances as FPs and 4 as FNs. The model achieves an accuracy of 99.3519% on adversarial data, slightly lower than the 99.8148% accuracy observed for normal testing data. This small difference highlights the effectiveness of the adversarial training process, as the model sustains high performance even in the presence of adversarial data.

When compared with the results for normal traffic classification, as discussed in the previous section, the confusion matrix shows a slight increase in false negatives, from 1 to 4, in the adversarial classification, while false positives rose from 1 to 3. Despite the adversarial nature of the input data, the model exhibits resilience, as evidenced by its consistently high accuracy and ability to accurately classify both normal and adversarial traffic packets.

Fig. 7 provides a comparative analysis of the model's performance metrics—including accuracy, precision, recall, and F1 score—on both adversarial and normal testing data. For adversarial traffic, the model achieved an accuracy, precision, and F1 score of 99.35%, along with a recall of 99.34%. In comparison, the model's performance on normal data yielded an accuracy, recall, and F1 score of 99.81% and a precision of 99.82%. The results show a slight decline in all metrics for adversarial data, with the differences in accuracy, precision, recall, and F1 score being less than 0.5%. Despite this minor reduction, the model maintains outstanding performance across all metrics.

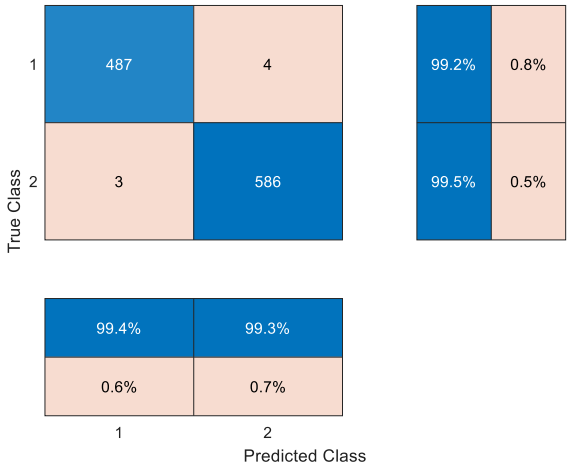

Figure. 6 The confusion matrix of evaluating the adversarial dataset using the proposed approach

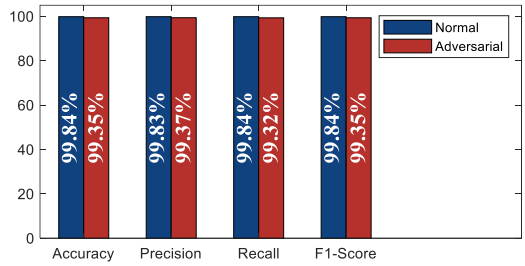

Figure. 7 The evaluation metrics bar chart of evaluating both normal and adversarial datasets using the proposed approach

These marginal drops indicate that the adversarial training strategy—specifically, the use of PGD adversarial training—has successfully enhanced the model's robustness to adversarial data. The findings confirm that the model performs comparably well on both adversarial and normal data, making it a reliable solution for real-world network traffic classification scenarios where adversarial data may exist.

7. Comparison

This section provides a detailed comparison of the proposed method with several state-of-the-art techniques for VPN traffic classification. Table 3 summarizes the key characteristics and performance metrics of these methods, highlighting their strengths and weaknesses.

Methods such as [11, 22] leverage statistical features extracted from network packets to classify VPN and non-VPN traffic. While these approaches are capable of handling variable-length packet flows, they do not incorporate all packet flow information, as they rely on summarized statistical features rather than processing the entire flow. In terms of accuracy, the method in [11] achieves 95.02%, with a recall of 97%, making it well-suited for detecting VPN traffic. However, its precision and F1-score are slightly lower, which may lead to imbalanced classification. Similarly, the method in [22] achieves an accuracy of 93.10% and employs temporal features across multiple dataset sizes, showcasing its adaptability to various scenarios. It should be noted that, for the sake of fair comparison, the binary classification scenario of [22] is only considered in this comparison. Despite these strengths, neither [11] nor [22] addresses adversarial robustness, limiting their utility in security-critical environments.

The method in [36] introduces Artificial Neural Networks (ANN) for VPN traffic classification, achieving an accuracy of 98.79%. This approach incorporates advanced feature extraction techniques to enhance the ANN's ability to differentiate between benign and malicious VPN traffic. While the method performs well in terms of recall (98.54%), it does not incorporate adversarial training and, like [11, 22], relies on summarized features rather than processing the complete packet flow. This limits its ability to capture the full dynamics of traffic, particularly in more complex or adversarial scenarios.

Deep learning-based methods, such as [35, 37], utilize trimmed packet flows of uniform size, which sacrifices the ability to handle variable-length flows in favor of standardized preprocessing. The method in [35], which combines ET-BERT and 1D-CNN, achieves 100% accuracy across all metrics for the binary VPN/non-VPN classification scenario, setting a high benchmark for normal traffic classification. However, it lacks robustness against adversarial attacks and does not incorporate all packet flow information. Similarly, [37], which integrates CNN and Swin Transformer, achieves 96.70% accuracy and an F1-score of 96.20%, outperforming several leading models in normal traffic classification. Nonetheless, its reliance on fixed-size flows and its inability to address adversarial scenarios are notable limitations.

The proposed method surpasses existing techniques by addressing critical gaps. Unlike [11, 22, 36], it is capable of processing variable-length packet flows while incorporating all traffic flow information without discarding relevant data. Moreover, it stands out as the only method in this comparison that integrates adversarial training (PGD), ensuring robustness against adversarial attacks. This dual capability enables it to achieve outstanding performance on both normal and adversarial traffic, with an accuracy of 99.81%, precision of 99.82%, recall of 99.81%, and F1-score of 99.81%. These metrics position it ahead of statistical feature-based

Table 3. The comparison of the proposed method with other network traffic classification approaches

Ref.	Method	Dataset	Adversarial attacks consideration	Can handle variable length packet flow	All packet flow information is incorporated	Accuracy	Precision	Recall	F1-Score
[11]	Statistical features and random forest	ISCX VPN-nonVPN	×	✓	×	95.02%	95.4%	97%	96.2%
[22]	Adaboost and light GBM	ISCX VPN-nonVPN	×	✓	×	93.1%	91.3%	94%	92.6%
[35]	ET-BERT and 1D-CNN fusion network (BCFNet) for encrypted traffic classification	ISCX VPN-nonVPN	×	×	×	100%	100%	100%	100%
[36]	Artificial Neural Networks (ANN) for VPN traffic classification	ISCX VPN-nonVPN	×	✓	×	98.79%	97.94%	98.54 %	96.84 %
[37]	Deep learning method combining CNN and Swin Transformer for encrypted traffic classification	ISCX VPN-nonVPN	×	×	×	96.7%	95.7%	97.3%	96.2%
Ours	EfficientNet-B0 combined with BiLSTM and PGD adversarial training	ISCX VPN-nonVPN	✓	✓	✓	99.81%	99.82%	99.81 %	99.81 %

approaches like [11, 22] and demonstrate competitive advantages over deep learning methods like [35, 37], particularly in handling dynamic and security-sensitive traffic scenarios.

When evaluating the trade-offs, it is evident that each method has its strengths. Approaches like [11, 22, 36] excel in handling variable-length packet flows but fall short in incorporating complete traffic information. Conversely, methods like [35] achieve perfect accuracy in specific scenarios but lack adaptability to variable-length flows and adversarial conditions. The proposed method effectively bridges these gaps by offering a balanced solution that excels in all major aspects: processing variable-length inputs, incorporating full packet flow information, and demonstrating robustness against adversarial attacks.

All in all, while each of the reviewed methods excels in certain areas, the proposed method distinguishes itself as a comprehensive and robust solution for VPN traffic classification. Its ability to handle diverse traffic scenarios, maintain resilience against adversarial attacks, and deliver superior performance metrics makes it a valuable contribution to the field.

8. Conclusion

This paper presents a method for network traffic classification, targeting the distinction between VPN and Non-VPN traffic. The method combines EfficientNet-B0, enhanced with biLSTM layers to preprocess raw network packet data. This preprocessing step is crucial for handling variable-length input sequences, allowing the network to automatically extract features without the need for clipping, padding, or manual feature extraction. The feature dimensions are expanded via fully connected

layers, which reshape the data for optimal processing through EfficientNet-B0's convolutional layers.

A key contribution is the introduction of Projected Gradient Descent adversarial training to enhance robustness against adversarial traffic patterns, such as network congestion or packet delays. The results demonstrate the effectiveness of this approach: the model achieves 99.81% accuracy on normal traffic and 99.35% on adversarial traffic. However, the results reveal some trade-offs. While PGD adversarial training significantly improves the model's resilience to attacks, a slight drop in performance is observed in adversarial scenarios compared to normal traffic. The model exhibits a small increase in false negatives and a minor decrease in metrics such as accuracy, precision, recall, and F1 score when evaluated on adversarial data.

These findings highlight the effectiveness of the proposed adversarial training technique while also pointing to areas for further exploration, such as tuning the balance between normal and adversarial data (λ parameter) to achieve better generalization across diverse conditions. The proposed method demonstrates strong potential, but further investigation can be done to improve robustness and ensure consistent performance under more challenging real-world traffic patterns.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Conceptualization, Thulfiqar Mahmood Tawfeeq and Mohsen Nickray; methodology, Thulfiqar Mahmood Tawfeeq and Mohsen Nickray; software, Thulfiqar Mahmood Tawfeeq; validation, Thulfiqar Mahmood Tawfeeq; formal analysis, Mohsen Nickray; investigation, Thulfiqar Mahmood Tawfeeq; resources, Thulfiqar Mahmood Tawfeeq; data curation, Thulfiqar Mahmood Tawfeeq; writing—original draft preparation, Thulfiqar Mahmood Tawfeeq; writing—review and editing, Thulfiqar Mahmood Tawfeeq and Mohsen Nickray; visualization, Thulfiqar Mahmood Tawfeeq; supervision, Mohsen Nickray.

Nomenclature	
X_k	The adversarial example at step k
$sign(.)$	Computes the sign of the gradient to determine the direction of the perturbation
Π(.)	The projection function
$\mathcal{U}(.)$	The uniform random distribution
ϵ	The maximum allowable perturbation for the adversarial examples
δ_0	The initial perturbation
g_k	Gradient of the loss function in kth iteration
$\mathcal{L}(.)$	The loss function
α	The step size
x'	The final adversarial example
$\mathcal{L}_{adv}$	Adversarial loss
$\mathcal{L}_{normal}$	Normal loss
λ	Adversarial rate parameter
f_θ	The output of the network with learnable parameters θ
x	The initial input of the network
y	The ground truth labels
∇_x	The gradient of the loss function with respect to the input
δ_k	The perturbation of the kth iteration
max	The maximum calculation operator
min	The minimum calculation operator
∇_θ	The gradient of the loss function with respect to learnable parameters
TP	True Positives
TN	True Negatives
FP	False Positives
FN	False Negatives
$\mathcal{L}_{Tot}$	The total loss
θ	The model learnable parameters
g_θ	The gradient of the total loss

References

[1] A. Azab, M. Khasawneh, S. Alrabaee, K. K. R. Choo, and M. Sarsour, "Network traffic classification: Techniques, datasets, and challenges", *Digital Communications and Networks*, Vol. 10, No. 3, pp. 676-692, 2024.

[2] R. H. Hwang, M. C. Peng, C. W. Huang, P. C. Lin, and V. L. Nguyen, "An unsupervised deep learning model for early network traffic anomaly detection", *IEEE Access*, Vol. 8, pp. 30387-30399, 2020.

[3] G. D'Angelo and F. Palmieri, "Network traffic classification using deep convolutional recurrent autoencoder neural networks for spatial–temporal features extraction", *Journal of Network and Computer Applications*, Vol. 173, 102890, 2021.

[4] S. Dong, "Multi class SVM algorithm with active learning for network traffic classification",

Expert Systems with Applications, Vol. 176, 114885, 2021.

[5] Y. Wang, X. Yun, Y. Zhang, C. Zhao, and X. Liu, "A multi-scale feature attention approach to network traffic classification and its model explanation", *IEEE Transactions on Network and Service Management*, Vol. 19, No. 2, pp. 875-889, 2022.

[6] R. Bozkır, M. Cicioglu, A. Calhan, and C. Togay, "A new platform for machine-learning-based network traffic classification", *Computer Communication*s, Vol. 208, pp. 1-14, 2023.

[7] J. Zhao, X. Jing, Z. Yan, and W. Pedrycz, "Network traffic classification for data fusion: A survey", *Information Fusion*, Vol. 72, pp. 22-47, 2021.

[8] S. M. Rachmawati, D. S. Kim, and J. M. Lee, "Machine learning algorithm in network traffic classification", In: *Proc. of the 2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1010-1013, 2021.

[9] M. Shafiq, Z. Tian, A. K. Bashir, A. Jolfaei, and X. Yu, "Data mining and machine learning methods for sustainable smart cities traffic classification: A survey", *Sustainable Cities and Society*, Vol. 60, 102177, 2020.

[10] Y. Yang, Y. Yan, Z. Gao, L. Rui, R. Lyu, B. Gao, and P. Yu, "A network traffic classification method based on dual-mode feature extraction and hybrid neural networks", *IEEE Transactions on Network and Service Management*, Vol. 20, No. 4, pp. 4073-4084, 2023.

[11] M. Al-Fayoumi, M. Al-Fawa'reh, and S. Nashwan, "VPN and Non-VPN Network Traffic Classification Using Time-Related Features", *Computers, Materials & Continua*, Vol. 72, No. 2, 2022.

[12] W. Zheng, J. Zhong, Q. Zhang, and G. Zhao, "MTT: an efficient model for encrypted network traffic classification using multi-task transformer", *Applied Intelligence*, Vol. 52, No. 9, pp. 10741-10756, 2022.

[13] M. Lotfollahi, M. Jafari Siavoshani, R. Shirali Hossein Zade, and M. Saberian, "Deep packet: A novel approach for encrypted traffic classification using deep learning", *Soft Computing*, Vol. 24, No. 3, pp. 1999-2012, 2020.

[14] M. M. Raikar, S. M. Meena, M. M. Mulla, N. S. Shetti, and M. Karanandi, "Data traffic classification in software defined networks (SDN) using supervised-learning", *Procedia Computer Science*, Vol. 171, pp. 2750-2759, 2020.

[15] I. A. Najm, A. H. Saeed, B. A. Ahmad, S. R. Ahmed, R. Sekhar, P. Shah, and B. S. Veena, "Enhanced Network Traffic Classification with Machine Learning Algorithms", In: *Proc. of the Cognitive Models and Artificial Intelligence Conference*, pp. 322-327, 2024.

[16] Z. Jin, Z. Liang, M. He, Y. Peng, H. Xue, and Y. Wang, "A federated semi‐supervised learning approach for network traffic classification", *International Journal of Network Management*, Vol. 33, No. 3, e2222, 2023.

[17] O. Aouedi, K. Piamrat, and D. Bagadthey, "A semi-supervised stacked autoencoder approach for network traffic classification", In: *Proc. of the 2020 IEEE 28th International Conference on Network Protocols (ICNP)*, pp. 1-6, 2020.

[18] S. Dong, Y. Xia, and T. Peng, "Network abnormal traffic detection model based on semi-supervised deep reinforcement learning", *IEEE Transactions on Network and Service Management*, Vol. 18, No. 4, pp. 4197-4212, 2021.

[19] A. Balachandran and P. P. Amritha, "VPN network traffic classification using entropy estimation and time-related features", In: *Proc. of IOT with Smart Systems: ICTIS 2021*, Vol. 2, pp. 509-520, 2022.

[20] 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.

[21] F. Hu, S. Zhang, X. Lin, L. Wu, N. Liao, and Y. Song, "Network traffic classification model based on attention mechanism and spatiotemporal features", *EURASIP Journal on Information Security*, Vol. 2023, No. 1, 6, 2023.

[22] G. Abbas, U. Farooq, P. Singh, S. S. Khurana, and P. Singh, "Feature engineering and ensemble learning-based classification of VPN and non-VPN-based network traffic over temporal features", *SN Computer Science*, Vol. 4, No. 5, 546, 2023.

[23] Z. Wang, Y. Feng, and H. Yan, "An effective real-time traffic classification method using convolutional neural network", *Research Square*, 2023.

[24] D. L. Romana, K. Sugitani, and Y. Musashi, "DNS Based Security Incidents Detection in Campus Network", *International Journal of Intelligent Engineering and Systems*, Vol. 1, No. 1, pp. 17-21, 2008.

[25] S. A. Mohsin and A. S. Alfoudi, "Internet Traffic Classification Model Based on A-DBSCAN

Algorithm", *International Journal of Intelligent Engineering & Systems*, Vol. 17, No. 5, 2024, doi: 10.22266/ijies2024.1031.72.

[26] S. R. Chikkalwar and Y. Garapati, "Autoencoder–support vector machine–grasshopper optimization for intrusion detection system", *International Journal of Intelligent Engineering and Systems*, Vol. 15, No. 4, pp. 406-414, 2022, doi: 10.22266/ijies2022.0831.36.

[27] M. Tan, "Efficientnet: Rethinking model scaling for convolutional neural networks", *arXiv preprint arXiv:1905.11946*, 2019.

[28] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks", In: *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7132-7141, 2018.

[29] C. Su and W. Wang, "Concrete cracks detection using convolutional neural network based on transfer learning", *Mathematical Problems in Engineering*, Vol. 2020, No. 1, 7240129, 2020.

[30] Y. Wang, X. Ma, J. Bailey, J. Yi, B. Zhou, and Q. Gu, "On the convergence and robustness of adversarial training", *arXiv preprint arXiv:2112.08304*, 2021.

[31] A. Mądry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks", *Stat*, Vol. 1050, No. 9, 2017.

[32] 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. of the 2017 IEEE International Conference on Intelligence and Security Informatics (ISI)*, pp. 43-48, 2017.

[33] P. Wang, F. Ye, X. Chen, and Y. Qian, "A robust and adaptive method for encrypted traffic classification", *IEEE Transactions on Information Forensics and Security*, Vol. 14, No. 11, pp. 2954-2969, 2019.

[34] 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. of the 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407-414, 2016.

[35] P. Zhu, G. Wang, J. He, Y. Dong, and Y. Chang, "An encrypted traffic identification method based on multi-scale feature fusion", *Array*, Vol. 21, 100338, 2024.

[36] S. A. A. Mohamed and S. Kurnaz, "Classified VPN network traffic flow using time related to artificial neural network", *Computers, Materials and Continua*, Vol. 80, No. 1, 819-841, 2024.

[37] Y. Wang, Y. Gao, X. Li, and J. Yuan, "Encrypted traffic classification model based on SwinT-CNN", In: *Proc. of 2023 4th International Conference on Computer Engineering and Application (ICCEA)*, pp. 138-142, 2023.