2021-09-21

Anomaly Detection in Encrypted Internet Traffic Using Hybrid Deep Learning

Bakhshi, T

10.1155/2021/5363750
Security and Communication Networks
Hindawi

Hindawi
Security and Communication Networks
Volume 2021, Article ID 5363750, 16 pages
https://doi.org/10.1155/2021/5363750

Research Article

Anomaly Detection in Encrypted Internet Traffic Using Hybrid Deep Learning

Taimur Bakhshi [1,2] **and Bogdan Ghita** [2]

[1]*Center for Information Management & Cyber Security, National University of Computer & Emerging Sciences, Lahore, Pakistan*
[2]*Center for Security, Communications & Networking Research, University of Plymouth, Plymouth, UK*

Correspondence should be addressed to Taimur Bakhshi; taimur.bakhshi@nu.edu.pk

Received 22 May 2021; Revised 30 August 2021; Accepted 1 September 2021; Published 21 September 2021

Academic Editor: Khizar Hayat

An increasing number of Internet application services are relying on encrypted traffic to offer adequate consumer privacy. Anomaly detection in encrypted traffic to circumvent and mitigate cyber security threats is, however, an open and ongoing research challenge due to the limitation of existing traffic classification techniques. Deep learning is emerging as a promising paradigm, allowing reduction in manual determination of feature set to increase classification accuracy. The present work develops a deep learning-based model for detection of anomalies in encrypted network traffic. Three different publicly available datasets including the NSL-KDD, UNSW-NB15, and CIC-IDS-2017 are used to comprehensively analyze encrypted attacks targeting popular protocols. Instead of relying on a single deep learning model, multiple schemes using convolutional (CNN), long short-term memory (LSTM), and recurrent neural networks (RNNs) are investigated. Our results report a hybrid combination of convolutional (CNN) and gated recurrent unit (GRU) models as outperforming others. The hybrid approach benefits from the low-latency feature derivation of the CNN, and an overall improved training dataset fitting. Additionally, the highly effective generalization offered by GRU results in optimal time-domain-related feature extraction, resulting in the CNN and GRU hybrid scheme presenting the best model.

1. Introduction

The ongoing development in the Internet of Things (IoT), data science, machine learning, and Internet services has resulted in a significant increase in network-related applications and services for end users. The continuous rise in user applications has also increased the number of threat vectors seeking to compromise the security and privacy of users [1, 2]. Applications are, therefore, increasingly relying on encryption schemes to provide adequate level of privacy and user confidence in Internet-based services. A consequence of encrypted traffic is the limitation of existing intrusion detection and prevention systems to analyze network traffic at the edge by Internet service providers (ISPs) and mitigate network-related attacks. Machine learning (ML) techniques have, therefore, seen increasing applicability in understanding (encrypted) network traffic patterns and providing anomaly inference capabilities. The simultaneous advancement in hardware and reduction in costs have furthered the analytical capability and scope of ML applications in several fields including network traffic analysis. Of particular interest is the deep learning (DL) paradigm, gaining further adoption in data science, developing artificial intelligence that an influence a wide array of research and development [1, 3]. Although still relatively nascent, deep learning primitives are being used to dynamically extract experience models from datasets generated under different environments and infer the underlying logic.

The applications of deep learning methods in network security are also gaining prevalence. Network administrators and service providers are researching and deploying state-of-the-art security solutions and are turning to deep learning and ML-based incorporation in their security devices. The use of sophisticated encryption techniques in network traffic

to enhance user privacy and security has also created an environment where hackers and miscreants are using the same traffic obfuscation techniques to launch cyberattacks [1, 4]. Although several studies have sought to classify and address network security threats, the analysis of encrypted traffic to identify malicious threats is an ongoing research challenge. Acquiring training data for the same is also a major problem in research studies. Encrypted data are usually trusted and are not decrypted to maintain data integrity and privacy. Encrypted traffic, however, if not inspected, can be exploited by attackers which can lead to successful network compromise(s) [1–3]. Deep learning technology offers a new perspective on the problem where data traffic does not require decryption, but the characteristics of the traffic are analyzed to deal with malicious traffic [1, 4]. The low latency offered by DL solutions, minimum human intervention, and timely response can allow respective solutions to be deployed for detecting attack traffic.

To this end, the present work proposes different encrypted traffic identification models using deep learning techniques to achieve high classification accuracy. The main contributions of this work are as follows:

(1) The present work uses multiple deep learning network structures independently as well as in hybrid assembly to assess and summarize the features of normal and encrypted malicious attack traffic. Each model uses its underlying algorithm to automatically adjust and improve weights for accurate classification while minimizing the loss function.

(2) A significant number of prior studies in DL-based traffic classification such as [5–7] have mostly relied on hand-crafted and manually intensive feature selection. The present work in comparison utilizes complex (raw) network traffic as input to dynamically extract fixed data characteristics. The approach allows greater automation in learning network data and discriminating abnormal behavior embedded in encrypted traffic by fully exploiting the feature selection for capability of DL structures.

(3) During validation, the proposed deep learning approach is tested on three publicly available datasets, namely, NSL-KDD [8], UNSW-NB15 [9], and CSE-CIC-IDS 2017 [10]. Employing different traffic datasets provides an accurate representation of application and service heterogeneity inherent in existing real networks for traffic classification purpose.

(4) The present work, therefore, compared to earlier efforts [6, 11–13], benefits researchers by undertaking encrypted traffic classification using DL techniques on a wider variety of publicly available traffic data, aiming to circumvent (any) inflated classification accuracy cofounded by the use of single or private datasets as well as hand-crafted input (traffic) features.

The rest of this paper is organized as follows: Section 2 gives the related work in utilizing deep learning technologies in cyber security. Popular neural network structures are overviewed along with a discussion of the most relevant literature review of deep learning approaches and their performance limitations. Section 3 provides the adopted methodology and specifics of the applicable neural networks in detail. Section 4 details the datasets and evaluation benchmarks. Section 5 analyses the experimental results. Section 6 draws final conclusions and comments on future work.

2. Related Work

The present section describes some of the recent advances in deep learning applications in the cyber security domain. Furthermore, some of the prominent studies related to present work, using DL approaches in network traffic classification for identifying network threats, are discussed.

2.1. Cyber Security and Neural Networks. Significant advancement in artificial intelligence made by the contributions of Hinton [4], LeCun [14], and Rosenblatt [15] has made deep learning one of the mainstream technologies. The developments in ML have led to several engineering and biomedical avenues employing deep learning to solve complex problems. The field of cyber security has no exception and is subject to evolving challenges that also aim to use DL algorithms for problem resolution [2, 16–20]. Deep learning, however, requires sufficient processing power, and the performance improvements of GPU architectures have helped in making cyber security-related DL applications on the ever-increasing Internet data volumes practical and fruitful [21, 22]. Research in academia, therefore, is paying more attention to the use of DL algorithms in facilitating robust cyber security. There have been several studies using deep learning in the security domain to research malware and threat mitigation, considered to be more effective than traditional machine learning procedures that may generate higher error rate and have limited ability to recognize masqueraded threats. The implementation scope of deep learning offers an added advantage, not relying on manual feature selection, allowing greater automation in finding hidden attack vectors. The applications are although nascent, and further perfection is needed to fully adapt deep learning technologies in the cyber security domain. The main deep learning structures continuing to gain traction in the cyber security field are convolutional neural networks (CNNs), deep belief networks (DBNs), auto encoding (AE), and recurrent neural network (RNN) with underlying variations in parameter selection and operational considerations [23, 24]. Not all deployments are, however, robust, and the DL structures also suffer from confidentiality issues where miscreants can exploit derived models to recover and investigate original input data [25].

Typical machine learning often presents higher requirements for feature derivation, and the efficacy of training and learning by the subsequent model(s) depends on the quality of extracted feature vectors. The learning model is, therefore, constructed based on feature vectors

[26]. For example, to detect botnets used in advanced persistent threat (APT) campaigns requires extraction of time, packet, and protocol related features to extract and train ML models [1, 23, 26]. In contrast, the feature engineering in deep learning is automatic saving manual efforts and generating high-quality models that can allow greater threat monitoring and classification accuracy. Malicious traffic that is encrypted presents an additional challenge to the security research community. Encrypted attack traffic (ET) can be identified using two methods presently available: (1) the recognition of attacks after decryption of network traffic and (2) nondecryption-based detection where traffic patterns are investigated and utilized to recognize threats. Production devices in service provider and user environments employ traffic classification for attack detection that would typically require significant computing resources to decrypt and (re)encrypt data in real-time affecting the network throughput, delay, and overall user experience. The costs of associated techniques such as deep packet inspection (DPI) are also substantial in terms of equipment and resources that these are not considered readily adoptable [27, 28]. The encryption and decryption of network traffic is not only resource intensive but also a challenging task for network managers in adequately preserving the privacy of end-users. Decrypted traffic logs may need to be stored on network storage that would be subject to privacy laws and compliance directives, making it even more difficult for service providers and network managers to contain information leaks and face regulatory penalties. Laws and regulatory concerns may also prevent the decryption of network traffic altogether [29]. The second approach is that of utilizing traffic patterns to identify attack vectors in encrypted traffic without performing any decryption. Encrypted traffic (ET) analysis provides an opportunity for research community and the industry to further the application of deep learning without compromising on data integrity or affect user experience [26, 29]. Encrypted traffic analysis using DL-based approaches, therefore, promises an important research direction explored in the present work.

2.2. Encrypted Traffic Analysis Using Neural Networks. Deep learning neural networks based on convolutional and recurrent algorithms have been gaining applicability in data science avenues over the past few years. Convolutional neural networks can be traced to biological preceptors associated with neurons capable of facilitating vision. The operation is based on a convolution-based algorithm and has been over time proved excellent in computer vision and image recognition. Studies have also used the CNN for text and natural language processing [22, 24]. The working principle of CNN describes a basic (convolution) layer being responsible for feature extraction and a pooling layer for minimizing features to obtain a generalization. Neuron numbers in the hidden layer(s) are directly proportional to the learning ability at the expense of greater computational cost. The time latency involved in feature extraction is relatively quick, relying on learning not only a single sample but creating associations between data. In a typical implementation, the first (few) convolutional layers determine and learn the simpler data features (e.g., edges and colour distribution in image processing) and the secondary layer(s) that is responsible for extraction of complex features. The complex features sometimes signify the correlations and relationships between the input data.

Recurrent neural networks are often employed to process ordered data in time, where special neurons can remember data that were previously processed [30]. Some back-propagation neural networks including CNN only consider input to output mapping without necessarily considering the order of an occurrence, i.e., the time dimension. Significantly, different from other network structures, the RNN includes memory function to consider input of the previous instance. The output, therefore, is a consequence of pre- and post-timing output and the overall structure employs feedback. If the data sequence is relatively long, then RNNs may suffer from short-term memory problem where there is latency and deviation in transferring signal(s) from an earlier time (iteration) to the latter. The gradient may disappear due to missing information at the beginning or become so small that learning can be suboptimal. Variations in RNN include the long-short term memory (LSTM) and gated recurrent unit (GRU) algorithms that seek to address the vanishing gradient and short-term memory issue [31]. RNNs are inspired by cognitive functions of humans and typically applied to handwriting recognition, while their applicability in traffic classification is not adequately mature at present.

Important contributions using different schemes of neural network structures in encrypted traffic classification and intrusion detection, along with performance caveats, are summarized in Table 1 and further discussed in the leading paragraphs.

An increasing number of challenges and opportunities in analyzing deep learning application modeling for threat monitoring and mitigation have been described by Zhang et al. [32]. The challenges can be categorically divided into the technical difficulties presented by effectively training neural network, feature processing issues, and acquiring labeled data in unsupervised learning. The research work highlighted the susceptibilities involved in countering threats and the suboptimal cooperative modeling to ensure sufficient privacy against attacks.

In some of the preliminary works, Wang et al. [5] tested convolutional neural networks (CNNs) for traffic classification. The study proved that the convolutional approach is effective in network traffic classification. However, whether the CNN structures can be used to identify attacks incorporated in encrypted and nonencrypted traffic was not specifically considered or any experimental evidence presented for the same. Complimentary works include one-dimensional (1D) CNN applied for encrypted traffic classification [7] on the publicly available ISCX VPN and non-VPN dataset [40, 41]. The method, however, used biased inputs limiting the implementation scope in wider network settings and only partially exploiting the automated feature selection capability offered by DL network structures.

RNN, specifically the long short-term neural network, was used by Ge et al. [34] to classify encrypted traffic (ET).

Table 1: Neural network-based traffic classification.

Network structure	Description	Traffic classification (normal and ET)						Performance parameters				
		ETC	FB	TD	UC	AE	LL	EF	VG	VY	UP	FW
Convolutional [5, 7, 22, 24, 32, 33]	Biological inspired preceptors	✓ Δ	X	X	IH	✓	✓ γ	l, h Δ γ	X	—	✓	—
Recurrent [31, 33, 34]	Representation in animal cognition	✓ Δ	✓	✓	IH	✓	—	l	✓	✓	—	—
Hybrid [6, 11–13, 35]	Hybrid structure (CNN, RNN, etc.)	✓ Δ γ	✓	✓	H	✓	✓ Δ	l, h Δ γ ∫	X	✓	—	✓ **Δ**
DPI + NN [33, 36]	DPI-facilitated ground truth	✓	—	—	H	—	X	h Δ	X	—	✓	✓
Miscellaneous [12, 37–39]	Multimodal approach	✓ Δ ∫	✓	✓	IH	✓	- ∫	h Δ ∫	—	✓	✓	✓ **Δ**
	Multitasking/inductive inference	✓ Δ ∫	✓	✓	IH	✓	✓	h Δ ∫	—	✓	✓	✓ **Δ**
	Federated models: locally trained	✓ Δ γ ∫	✓	✓	IH	✓	—	l, h Δ γ	—	- ∫	✓	✓ **Δ**

Note. LL: low latency; ETC: encrypted traffic classification capability; FB: feedback mechanism; UC: use-case in (1) isolation (I) and (2) hybrid (H); AE: autoencoding; EF: efficiency, 1, high (h); 2, low (l); and 3, diluted (d); VG: vanishing gradient; VY: algorithm variety; UP: database update required; FW: further investigation needed; Δ: testing on limited or private dataset or applications; γ: statistical or manually engineered input features resulting in inflated classification accuracy; ∫: black-box algorithms requiring further testing of input datasets (with parameter tuning) on several models.

The work used long short-term memory (LSTM) to account for sequential characteristics in network traffic. While the system promised high accuracy in relatively smaller private network environments, the feature itself is not considered scalable and can be diluted in classifying traffic in larger networks. Lopez-Martin et al. [6] proposed a combinatorial scheme comprising of LSTM and two-dimensional (2D) CNN. The approach was evaluated on traffic available from an academic network and showed high accuracy (>90%). The authors illustrated penalties that are associated with using packet interarrival times along with port numbers as an input feature. The work, however, again used inputs that comprised of flow-level statistical information, ports, time-series feature vector, and deep packet inspection (DPI) for class labeling of ground truth data. The approach, therefore, only partially utilized the DL benefit of automating feature selection and significantly relied on manually crafted input features that bias the classification accuracy. Moreover, encrypted traffic classification and intrusion detection was not specifically considered in the evaluation.

Liu et al. [13] also utilized the recurrent neural network in tandem with a flow sequence network (FS-Net) for encrypted traffic classification. Using multilayer and bidirectional RNN (encoder and decoder), the FS-Net automatically learns classifying features from raw traffic flows eliminating the need for manual selection. The decoder layer (GRU) uses reconstruction to boost the discriminatory characteristics between traffic flows. A campus traffic dataset [42] comprising of a limited number of applications (18) was employed for testing purposes reporting a high accuracy (99%). Despite the significantly high accuracy, dataset limitations and focus on certain applications require further benchmarking and scalability evaluation of FS-Net on a wider dataset comprising diverse network traffic.

Similarly, Zeng et al. [35] also proposed the use of DL for automated feature selection from raw traffic for models: CNN, LSTM, and stacked autoencoding (SAE). The CNN was used to learn spatial features, while LSTM focused on time-related aspects and SAE to capture coding characteristics. Two datasets comprising of VPN [40] and non-VPN [41] traffic were used for evaluation. The traffic classification and intrusion detection accuracy of DFR surpassed conventional decision tree ML methods (C4.5, KNN) as well as 1D-CNN models on the preprocessed datasets. While autoencoding (AE) aids in reducing the dimensionality of input data, complexity of relationship between the dataset and features which is quite common in ET classification problem can also result in imperfect decoding and misunderstanding of important data variables. An additional caveat is the use of stacked autoencoding to circumvent imperfections in dimensionality reduction at the cost of higher processing times.

In addition to conventional networks, mobile network traffic presents a unique challenge for network providers due to increasing demand for offering value-added services according to user demand and an evolving traffic mix, while privacy concerns require accurate classification and filtering of encrypted traffic. Aceto et al. [11] discussed the fusion of DL techniques such as stacked autoencoding (SAE), CNN, and LSTM for traffic classification in mobile networks. Using mobile traffic data from three (human) users, incorporating several Android and iOS applications, a performance comparison of multidimensional CNN, shallow ML, and CNN + LSTM was evaluated. 1D-CNN reported highest accuracy for Android and 2D-CNN for iOS systems, respectively. In a similar work [12], multimodal dependencies between the traffic classification objects are utilized to exploit sophisticated DL layers (inception and residual neuron connections) increasing the efficiency of traffic classification for iOS applications using heterogenous input data. It was observed that pretraining can lead to improved transfer learning by using unsupervised data and the ability to cope with challenges such as *traffic bursts* for different input types. Overall, in these complimentary traffic classification approaches, the researchers were of the view that recognition of hyperparameter importance in DL algorithm implementations along with the use of hybrid architectures requires further investigation [11, 12].

An extrapolation of the same multimodal approach considered in [12] was investigated in the context of multitasking [37]. Multitasking refers to an inductive transfer to boost learning in one task using the information contained in training (signals) of related tasks [43, 44]. Multitask learning may yield a single comprehensive model using

parallel execution reduces computational overhead while building a classifier that can offer better generalization. The time complexity and overall accuracy when tested on wide variety of datasets may require further evaluation to appreciate the level of visibility such generalization can allow. Huang et al. [38] also used multitask learning to solve malware detection and intrusion detection using the 2D-CNN model on the dataset obtained from CTU-13 [45] and VPN [40, 41]. However, similar to [7], biased inputs were used that may limit the applicability of proposed approach on complex data and not fully appreciate the significance of DL in automating feature extraction.

Model building using federated learning has been proposed by Zhao et al. [39]. To address privacy concerns, model learning is carried out in a distributed way where local trace data are not shared, but only the learnt models to generate a complete model. The use of statistical features using bidirectional flows again only partially utilizes the feature learning capability offered by DL algorithms, biasing the input for certain neural network models. The performance improvements due to federated learning using in [39] although reported a satisfactory increase in accuracy along with reduction in training time as compared to nonfederated approaches.

Mohammad et al. [36] used deep packet inspection (DPI) in tandem with neural networking to detect the encrypted network traffic. The results showed highly promising discrimination between encrypted and non-encrypted traffic. However, the detection and recognition of encrypted attack traffic from normal traffic was not distinctively considered. Surveys such as Rezaei et al. [33] categorize at length the application of deep learning algorithms in various encrypted traffic classification scenarios. DPI in general provides the best detection accuracy albeit processing cost and difficulty in relying on building and continuously updating raw payload patterns to classify encrypted traffic.

Industry and academia are recognizing deep learning as an important area offering prospects for the network security arena [5, 33]. Deep learning is used to classify network traffic, but also it allows researchers to develop methods for accurate and timely detection of attacks embedded in Internet application traffic. While the implementation of deep learning frameworks has allowed for greater reliability in traffic classification, research on encrypted traffic threat detection using DL technologies is relatively nascent. It is also noted the black-box nature of most DL algorithms, where the impact on accuracy due to specific selection of inputs is not predictable and remains an open challenge [11]. To understand binary data as well as perform flow-based security analysis, the viability of deep learning methods still needs to be investigated and benchmarked across different network environments. Existing approaches offer limited maturity in applying DL structures to achieve unbiased conclusions where the reported accuracy is not inflated due to data input [37]. A significant number of previously proposed solutions are limited by (1) the variety of datasets on which these are tested to prove classification/intrusion detection viability real network environments and (2) ignoring a fundamental aspect of DL-based classification which is to allow the use of raw input in contrast to manual feature selection and achieve higher classification accuracy while reducing human intervention.

The present paper employs deep learning to aid in the automated extraction of precise features, allow greater dimensional analysis of (network) data, and extract the feature vectors pointing to anomalies, focusing on four publicly available datasets from a wider mix of network traffic. The paper uses neural networks, including convolutional, recurrent, and shallow network structures preceded by preprocessing of data to optimally train model(s). The proposed methodology is considered in detail in the following section.

3. Proposed Methodology and Neural Network Structures

In the present work, different deep learning methods are used to analyze the attack detection capability on encrypted traffic. The first algorithm used is CNN allowing low-latency training and providing the weightage(s) after initial parameters are set. CNN reduces high-dimensional features using maximum pooling to determine the most characteristic features needed. Subsequently, the RNN algorithm is employed to extract the time-series-related characteristics of use in small-scale network environments. During validation, the same testing data will be used to determine the efficiency of different models, aiming to combine the benefits and develop a new hybrid model.

A brief background of the network structures, along with detailed construction specific and illustrations, is discussed in the following sections.

3.1. Convolutional Neural Network (CNN). The convolutional layer kernel is like a filter functioning to extract the most important feature(s) from input data. Each convolution of input data results in a feature graph, where each convolution kernel provides a different feature. The next layer in turn is pooling used to compress features and select the most applicable. As layers increase, the number of parameters also changes exponentially. Having too many parameters may affect the performance of the network. The network culminates in a final connectivity layer at the end to combine and categorize the results. The overall error rate is predicted using loss functions, cross entropy, etc. The output depends on three parameters: convolutional kernels directly proportional to the volume of output, short and long step size dictating the amount of data processed at a time, resulting in fine grained or coarse feature selection, and filling to keep learning step constant and smooth. The convolution operation uses the following formula:

$$B(i, j) = \sum (m = 0) \sum (n = 0) K(m, n) \cdot A(q - m, j - n). \tag{1}$$

A total of 4 convolutional layers are used in the present study, and the size of kernel is 3×3, which is able to extract the relevant features. The first layer has 32 neurons, containing fewer parameters and used to process data initially.

Each of the subsequent two layers has 64 neurons to improve model learning. The aim of the next dropout layer is to improve the overfitting problem as best as possible. Feature filtering is done by the maximum pooling layer (desired and differentiated features). The convolutional layer processes final features that are compressed and mapped by the final dense layer. The specifics and parameters for the convolution network used in the present study are given in Figure 1.

3.2. Recurrent Neural Network (RNN). In the RNN model, the neurons in the same layer are connected to each other and the output of previous (hidden) layer will be considered in calculating the input of the next layer similar to the input layer. To cater for a greater understanding of time-series traffic classification and threat analysis, RNN may offer a special ability [3, 33], shown in Figure 2. The processed data (traffic) are stored and referenced while processing the present data affecting the result. However, as mentioned earlier, direct application of RNN may suffer from gradient disappearance.

Gated recurrent unit (GRU) and long short-term memory (LSTM) may solve the short-term memory issue by keeping track of dependencies over a longer run for the input. The GRU uses two vectors: update gate and reset gate that are trained to decide and keep the information that needs to be provided to the output. The update gate determines the amount of information (past) that would require to be sent to the future partially or wholly and thereby eliminate the vanishing gradient issue. Similarly, reset gate is used to determine the amount of past information that needs to be forgotten. LSTM, in contrast, aids in filtering the previous states based on RNN, so that previous states having greater influence of the present can be selected, rather than just using the latest [8]. Although there can be variations in the implementation of GRU and LSTM network, however, the sigmoid activation function is usually considered as best in filtering signal(s) and producing output through a tangential (TanH) function [31, 46].

In the present work, we employ LSTM- and GRU-based models having a similar network structure. The input is fed to the first layer of an RNN followed by the dropout layer, adding uncertainty to the network. The hidden layers incorporate both GRU and LSTM neurons. GRU is more in number because LSTM neurons employ a greater number of training parameters. Excessive neurons will reduce training speed. Reduction in LSTM neurons in the hidden layer still outnumbered those for GRU. The final layer in the two models will be a dense layer to process the data. The formulae for GRU and LSTM are provided in (2)–(4) and (5)–(8), respectively. Common nonlinear activation functions sigmoid and rectified linear unit (ReLU) by (9)–(12), respectively [47]. The corresponding illustrations are provided in Figure 3:

Gated recurrent unit (GRU) model input and variables.

$$r_t = \text{sigmoid}\left(W_r\left[h_{t-1}, x_t\right] + b_r\right), \tag{2}$$

$$z_t = \text{sigmoid}\left(W_r\left[h_{t-1}, x_t\right] + b_z\right), \tag{3}$$

$$r_t = \text{sigmoid}\left(W_r\left[r_t \cdot h_{t-1}, x_t\right] + b_c\right). \tag{4}$$

Long short-term memory (LSTM) model input and variables are as follows:

$$i_t = \sigma\left(W_i\left[h_{t-1}, x_t\right] + b_i\right), \tag{5}$$

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_c\right), \tag{6}$$

$$f_t = \sigma\left(W_f\left[h_{t-1}, x_t\right] + b_f\right), \tag{7}$$

$$C_t = \tanh\left(W_C \cdot \left[h_{t-1}, x_t\right] + b_C\right). \tag{8}$$

Sigmoid function is as follows:

$$g(z) = \frac{1}{\left[1 + e^{-z}\right]}, \tag{9}$$

$$g\prime(z) = g(z)(1 - g(z)). \tag{10}$$

Rectified linear unit (ReLU) function is as follows:

$$g(z) = \max(0, z), \tag{11}$$

$$g\prime(z) = \begin{cases} 1, & z > 0, \\ 0, & \text{otherwise.} \end{cases} \tag{12}$$

The general loss function corresponding to weight adjustments is given by

$$J(W) = \frac{1}{n}\sum_{i=1}^{n} L\left(f\left(x^{(i)}; W\right), y^{(i)}\right). \tag{13}$$

For binary classification problems such as the prediction between normal and malicious traffic, binary cross entropy loss (softmax function) [48] can be used as given in

$$J(W) = \frac{1}{n}\sum_{i=1}^{n} y^{(i)}\log\left(f\left(x^{(i)}; W\right)\right) + \left(1 - y^{(i)}\right)\log\left(1 - f\left(x^{(i)}; W\right)\right). \tag{14}$$

In (13) and (14), $y^{(i)}$ is the actual while $f(x^{(i)}; W)$ is the predicted value.

4. Dataset and Evaluation Metrics

The present section discusses the datasets and the evaluation metrics used to benchmark the efficacy of proposed deep learning techniques for encrypted traffic classification.

4.1. Datasets. As discussed earlier, while reviewing the existing literature, using singular datasets may not comprehensively represent existing real networks. To provider an effective benchmark of the proposed methodology, three different datasets, namely, the NSL-KDD [49], UNSW-NB15 [9], and CIC-IDS-2017 [10], are employed. The number of records in each of the tested datasets is sufficient without the need to randomly select smaller portions in evaluating the consistency and comparability of our proposed structure in different network environments. Prior studies have widely

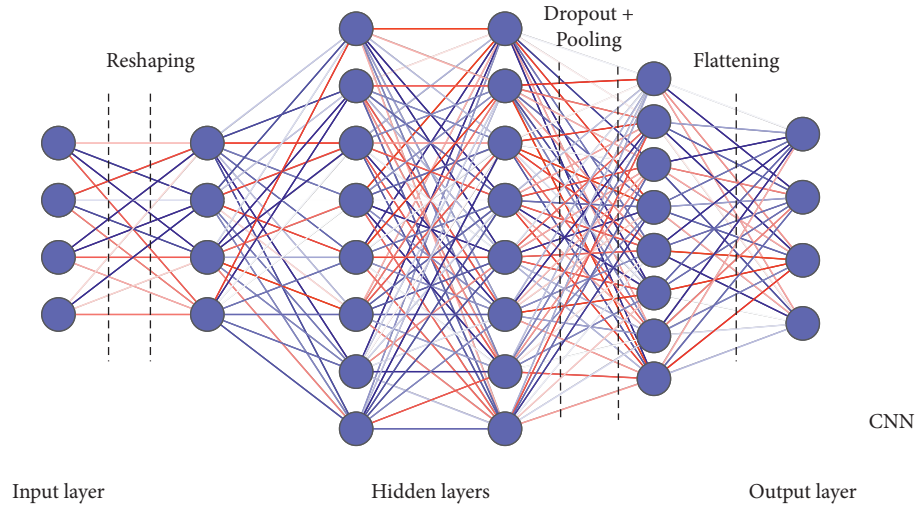

Figure 1: Structure of proposed CNN.

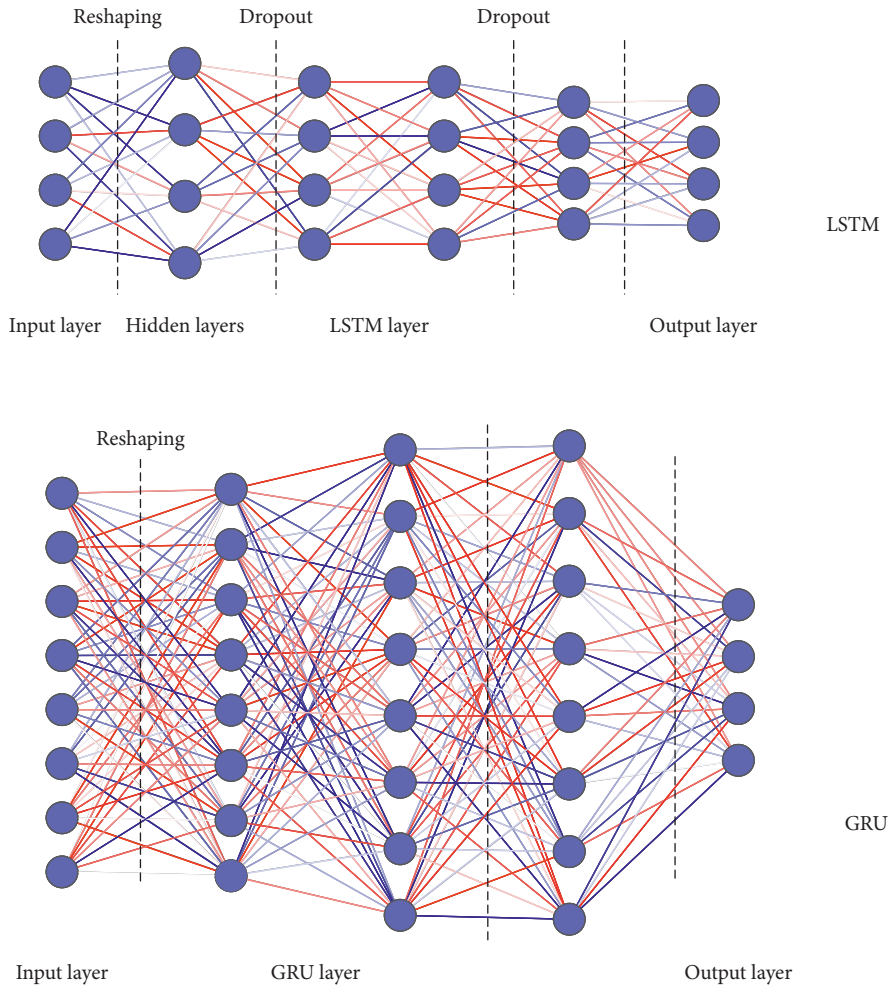

Figure 2: Structure of proposed RNN structure: (a) LSTM and (b) GRU.

used the NSL-KDD dataset for cyber security research [49, 50]. It is an N revised version of the original KDD99 dataset. There are a total of 148,517 record samples, each having 42 feature values available in the NSL-KDD dataset [8, 49]. UNSW-NB15 [9] is substantial in comparison having a total of 2,540,043 record samples, each having 48 features. CIC-IDS-2017 [10] comprised of 83 traffic features, having a total of 2,827,263 records. To address scalability concerns,

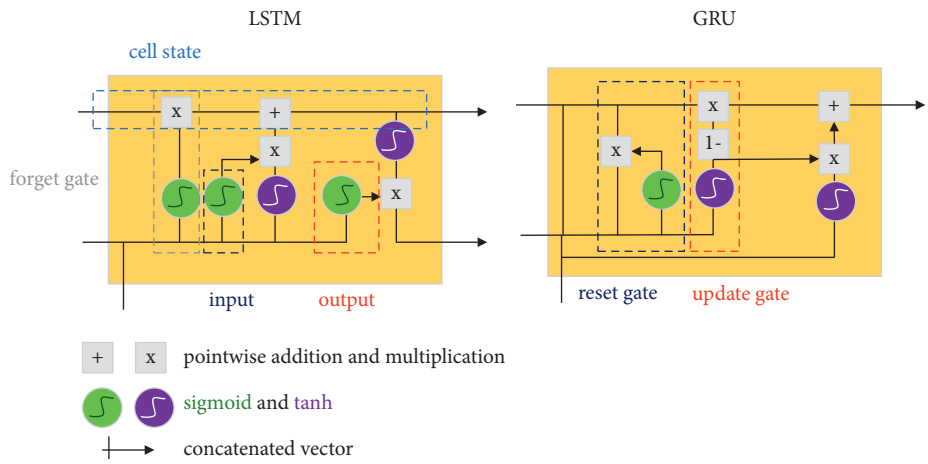

Figure 3: RNN internal structure.

we will not describe the individual features of each dataset and would refer the reader to the original source [9, 10, 49].

In the present study, the labeled data from each of the datasets will be sanitized according to binary classification and changed for subsequent experimentation. Binary classification requires distinguishing between encrypted malicious and other traffic. Focus will be on protocols used in the original dataset and relabeling the attack traffic over *https* and *ssh*, and any other encrypted protocol (*tls*, *ftp*, etc.) that can be used to carry attack traffic. Although encrypted traffic is a relatively small part of the three different datasets used in this work according to the summary information provided in Table 2, it conforms to real-world network environments. The data will be processed as follows: raw data will be processed using string updating, normalization followed by one hot coding before being used. The categories of binary and 4-class labeling of the resulting data for NSL-KDD, UNSW-NB15, and CIC-IDS-2017 and corresponding training and testing records are presented in Table 2(a), 2(b), and 2(c), respectively. The training and testing split for all datasets has been selected according to earlier findings [13, 35, 44] ensuring that the training and testing sets provide sufficient representation of encrypted and nonencrypted attack traffic. A visual description of traffic distribution in each of the datasets is given in Figure 4. Normal traffic (nonencrypted and encrypted) comprises bulk of the data in each traffic set (~50–85%). It is also notable that nonencrypted attack traffic is considerably substantial (~11–56%) in comparison with encrypted malicious traffic constituting only a minor part of each traffic set (~0.12–1.28%). The above traffic distribution highlights the relatively smaller percentage of encrypted attack traffic in overall network data in realistic network environments, making classification of malicious ET challenging as well as necessary to mitigate threats.

A further *subapproximation* of different protocols (*tls*, *ssh*, *http*, *ipps*, *ldaps*, *ftp*, etc.) constituting malicious traffic in the nonencrypted and encrypted categories for each dataset is provided in Figure 4. While the datasets represent variation in traffic volume (percentage) distribution among the above protocols, *tls* and *ssh* are most frequent across the

Table 2: Datasets and features.

Distribution	Traffic type		Training data	Test data
	Encrypted	Nonencrypted		
(a) Dataset and features (KDD) [8, 49]				
2-Dimensional	Normal	—	2344	1636
	Attack	—	1620	134
4-Dimensional	—	*Normal*	**64896**	**8088**
	—	*Attack*	**57002**	**12707**
	Normal	—	**2344**	**1636**
	Attack	—	**1620**	**134**
Total	—	—	125862	22565
(b) Dataset and features (UNSW-NB15) [9]				
2-Dimensional	Normal	—	25109	22163
	Attack	—	2178	1947
4-Dimensional	—	*Normal*	**1462481**	**709382**
	—	*Attack*	**221639**	**95144**
	Normal	—	**25109**	**22163**
	Attack	—	**2178**	**1947**
Total	—	—	1711407	828636
(c) Dataset and features (CSE-CIC-IDS 2017) [10]				
2-Dimensional	Normal	—	39793	28187
	Attack	—	2119	1215
4-Dimensional	—	*Normal*	**1505112**	**603724**
	—	*Attack*	**469227**	**177886**
	Normal	—	**36793**	**28187**
	Attack	—	**5119**	**1215**
Total	—	—	2016251	811012

encrypted attack traffic. For NSL-KDD, *ssh* protocol represents approximately ~30–60% of all encrypted attack traffic. In UNSW-NB15, a similar concentration of *ssh* (~10–15%) and *tls* (~20–28%) is recorded in the encrypted malicious traffic category. Similarly, in CIC-IDS-2017, *ssh* and *tls* usage in encrypted attack traffic amounts to around (~20–40%) each. Some of the other notable protocols in UNSW-NB15 and CIC-IDS-2017 include *http*, *ftp*, *ipp*, and *ldap* (and minor percentage of *snmp*, *smtp*, *telnet*, *dns*, etc. categorized as others) in encrypted and nonencrypted mode. The training and testing data in each protocol category for encrypted and nonencrypted attack traffic were split in approximately equal proportion as depicted in Figure 4. The

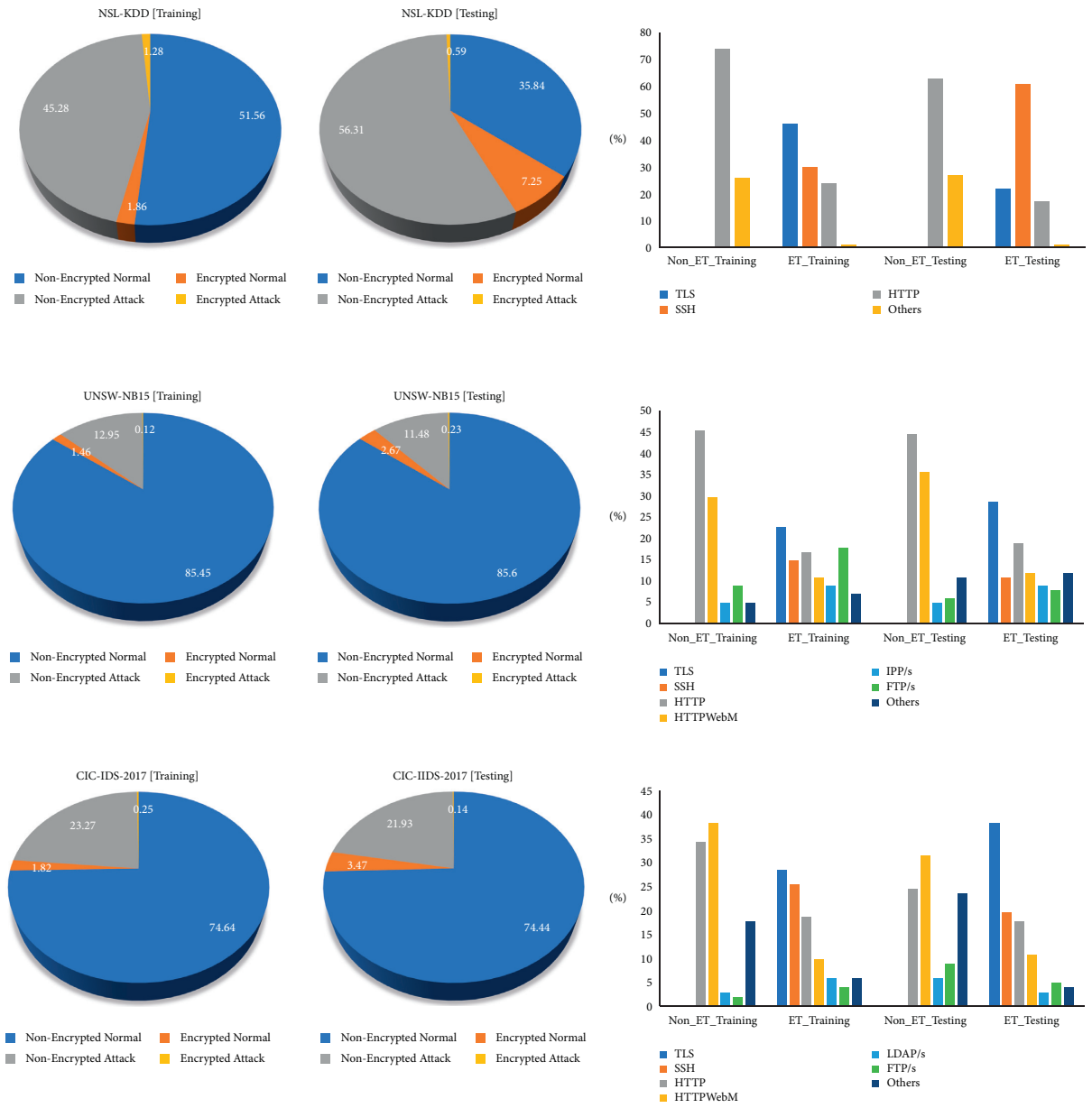

Figure 4: Datasets and attack traffic composition.

aim was to ensure that sufficient volume (percentage) of each protocol/attack vector is represented to account for common and uncommon traffic objects and optimize the overall classification accuracy.

4.2. Evaluation Metrics. The proposed neural network models are evaluated on the derived dataset to compute the accuracy, recall value, precision, false positive, and false negative rate.

True positive (TP) and true negative (TN) identify the (attack) samples correctly identified as belonging to anomalies and normal samples, respectively. False positive (FP) refer to the sample incorrectly identified as attack and false negative (FN) the samples incorrectly identified as normal traffic. The evaluation metrics are detailed in Table 3.

5. Experiment, Results, and Discussion

The present section details the specifics of the experiment, recorded results, and respective discussion of the best model.

5.1. Performance Evaluation. As mentioned earlier, three different datasets are used with respective features after normalization. Multiple feature set (4-dimensional scheme) was used for classification over the relatively basic two-feature labeling to fully analyze the efficacy of tested models on each dataset.

We employ the model structure provided in Figures 1 and 2 for performance evaluation of all traffic (data) sets. The CNN model consists of 1 input layer having 64 neurons and 3 convolutional layers (again each having 64 neurons) in the

Table 3: Evaluation metrics description and calculation.

Metric	Description	Calculation
Accuracy	The ratio of number of samples correctly identified divided by the total number of samples in the provided data test.	Accuracy = (TP + FP)/(TP + FP + TN + FN)
Precision	Defines the positive predictions that were correctly identified by the classifier.	Precision = (TP)/(TP + FP
Recall	The ratio of number of attacks correctly classified divided by the total number of attack samples.	Recall = (TP)/(TP + FN)
$F1$-score	The metric is two times the mean value determined for precision and recall computation.	$F1$-score = 2 × (precision × recall)/(precision + recall)
False positive (FP) rate	The metric is the ration of attack samples to normal samples.	FPR = FP/(FP + TN)
Normalization factor	The dataset is subjected to normalization as given in Figure 4. Where data values are greater than 1, all mappings are normalized between 0 and 1, to increase accuracy. The calculation method uses maximum and minimum values of input data column and then to each data.	$Z = (z\text{–min})/(\max\text{–min})$

hidden layers, as well as the maximum pooling, dropout, and flattening layer. Dropout argument is set to the typical value of 0.5, and the activation function used is rectified linear unit ReLU [22, 24, 47]. The output is based on FATE layer (federated AI) [51]. The output is classified using softmax activation function [48]. The LSTM model consists of 1 input layer (64 neurons), 3 LSTM layers (each having 64 neurons), and dropout layers in the hidden layer. The dropout is again 0.5, activation function is ReLU, and output is FATE layer classified by softmax. Similarly, the GRU model consists of 1 input layer (64 neurons), 3 GRU layers (having 64, 128, and 138 neurons), and dropout layers in the hidden layer. The dropout is set to 0.5, and output is again FATE using softmax activation. As highlighted in the discussion on neural network structures, the GRU has overall a greater number of neurons than LSTM; however, the actual number of parameters is smaller.

The control variables are listed in Table 4, and the same criteria are used for each of the datasets independently. The epoch is set to 10 using trial and error. The input as well as the output dimensions are the same. The batch size uses the same number of classified classes, with size and number of categories also kept as the same. The classified optimizer is based on Nadam (Nesterov-accelerated Adaptive Moment Estimation) gradient descent algorithm to minimize the cost function [54]. Nadam accelerates model learning by summing the exponential reduction of moving averages for present and previous values of the gradient. Nadam is available in Keras and has been successfully used in prior traffic classification studies to reduce cost [55]. Similarly, the loss function used in the present study and provided by Keras is *categorical_cross_entropy* [44] and given in (14). The objective function (loss function) provided in (13) and (14) is used to compute the loss quantity that any given neural network model would seek to minimize in the training phase.

The experiment was carried out using system features specified in Table 4. The hardware specification comprised of 64 GB RAM DDR4, with an Intel ® Core i7-7700K CPU at 4.20 GHz. The operating system used was Ubuntu 18.04.5 LTS (64 bit). The test platform was built using Keras 2.4.0 (Python-based deep learning API) [52] along with TensorFlow library version 2.2.2 [53].

The efficiency of each model (per dataset) is reflected in Table 5, for the same classification/parameters. Overall, the training speed of the CNN model is minimum compared to the rest while reporting high accuracy and low failure. Other models are subsequently added to the testbed to compare and determine the optimal model.

5.2. Time Complexity. We analyze the run time of each of the deep learning models, focusing on the training time as an indicator of operational value. The summarized performance for each model according to the three different datasets is provided in Table 5. Each model was subjected to 10 training rounds, on each of the three datasets, each having the same initialization parameters. For the NLS-KDD, the CNN model processed 148,634 samples in about 19.34 s for each round, and the total training time was 170.6 s. The LSTM model completed the same in approximately 85 s for each round and a total training time of 856.8 s. For the GRU model, the training time was 88 seconds for each round, and a total completion time was 878.2 s. The training time of shallow learning is quite small, each round consuming approximately 2 s and total time duration of 16.54 s. The UNSW-NB15 and CIC-IDS-2017 datasets provided similar results albeit a variation in the overall processing time as the respective sizes of the traffic sets was substantially greater than the NLS-KDD. UNSW-NB15 training time for CNN was noted at 185.14 s while 215.61 s for the CIC-17, a minimum increase when compared to CNN training in NSL-KDD despite the greater number of samples processed. The LSTM model training time was significantly higher at 2516.7 s and 2981.2 s, respectively, for NB15 and CIC-17. The training time for the GRU model was 1515.2 s and 1425.3 s for the same. Shallow models continued to present the minimum training time. Unlike CNN structures, RNN models intend to build a relationship between the hidden vectors at each step resulting in an overall increase in latency [56]. Although LSTM and GRU allowed the modeling of sequential traffic data from each of the three datasets, interdependence between outputs from consecutive timesteps limits parallelism resulting in an increase in processing time for longer sequences.

Table 4: Experiment parameters.

Specification	Parameters	Comments
Hardware	Core i7 7700K 8 MB 4.20 GHz 64 GB RAM DDR3L 1333	—
Software	Ubuntu 18.04.5 LTS 4.15.0–135-generic x86_64 bit	—
	Keras 2.4.0.	Python-based deep learning API [52]
	TensorFlow 2.2.2.	Open-source machine learning platform/library [53]
Neural network *Convolutional/Recurrent*	Activation function ReLU	ReLU is a piecewise linear function, used as the default for many neural network implementations and given in (11) and (12) [47]
	Final layer = FATE/FederatedAI	Federated AI framework is prevalent in deep learning neural networks, specifically at the final layer [51]
	Dropout = 0.5	Dropout is a regularization method used to learn a fraction of weights in the network per iteration and resolves common overfitting issue. Typical value used in large networks to achieve maximum regularization is 0.5 [22, 24].
	Output = softmax function	Softmax is a normalized exponential unction, often used as the last activation in neural network layers to normalize the output [48]
	Optimizer = Nadam	Nadam is a gradient descent algorithm frequently employed to minimize the cost function [54, 55]
	Loss function = categorical_cross_entropy	Categorical cross entropy is a loss function aiding in the computation of quantities that neural network model(s) aim to reduce in the training phase [44]

Table 5: Deep learning model performance.

Performance	Traffic Dataset	Network structure				
		Shallow	CNN	LSTM	GRU	Hybrid: CNN + GRU
Training duration (s)	NSL-KDD	**16.54**	170.6	856.8	878.2	375.6
	NB15	**27.41**	185.14	2516.7	1515.2	595.61
	CIC-17	**35.14**	215.61	2981.2	1425.3	741.42
Testing duration (s)	NSL-KDD	**1.5**	2.51	4.34	3.13	3
	NB15	10.1	**6.1**	18.9	20.5	16.4
	CIC-17	32.1	**9.2**	19.3	18.1	17.3
Accuracy (%)	NSL-KDD	78.51	**93.56**	79.99	92.01	93.10
	NB15	76.14	85.21	68.12	85.74	**91.21**
	CIC-17	68.19	86.92	71.15	80.07	**90.17**
Precision (%)	NSL-KDD	84.12	**94.61**	76.37	93.52	93.75
	NB15	81.57	89.17	78.51	81.57	**91.19**
	CIC-17	69.25	90.84	72.46	88.61	**92.34**
Recall (%)	NSL-KDD	91.51	89.99	80.24	91.12	**90.29**
	NB15	85.61	91.16	81.52	88.56	**91.36**
	CIC-17	78.52	90.2	89.12	85.45	**91.24**
FPR (%)	NSL-KDD	1.99	1.56	13.14	7.01	**1.12**
	NB15	2.88	1.89	10.25	7.63	**1.58**
	CIC-17	5.68	2.35	15.85	8.91	**1.57**
*F*1-score (%)	NSL-KDD	88.98	91.76	79.88	91.62	**92.34**
	NB15	85.91	92.8	81.51	93.21	**93.61**
	CIC-17	86.35	**93.56**	80.25	91.27	92.05

The hybrid CNN + GRU model reported a total training time of approximately 375.6 s for NSL-KDD, 595.61 s for NB15, and 741.42 s for CIC-17. The overall training duration of the hybrid model is relatively greater than that of CNN and significantly lower than that recorded for LSTM and GRU models, offers the inherent advantages of faster feature extraction (CNN) while also considering high dimensionality in features allowed by the GRU neural network.

During testing, all 22,544 samples in NSL-KDD can be validated in seconds. The CNN model reported 2.51 seconds, LSTM 4.34 seconds, and GRU 3.13 seconds, with the shallow model 1.5 seconds. CNN and GRU combination consumed approximately 3 seconds in testing. The CNN model testing on NB15 and CIC-17 recorded a two- and three-fold increase approximating at 6.1 s and 9.2 s, respectively. Similarly, the testing time for the hybrid model was 16.4 s and

17.3 s for NB15 and CIC-17 due to a greater volume of testing samples. The testing time for shallow, LSTM, and GRU models was greater than that of the CNN and the hybrid model for each of the datasets. Although the execution time for DL models is relatively high compared to typical shallow and regression-based models due to training with larger datasets, once trained the testing phase takes lesser time. Since deep neural networks can reduce data dimensionality and learn the most critical features, the models such as hybrid (CNN + GRU) structure can be used in real network environments with relative ease.

5.3. Discussion and Comparison. A comparison is made between the different models, and results of the accuracy, precision, recall, and *F*-1 score are demonstrated in Figure 5. To determine the optimal model, CNN and RNN structures were used for each of the datasets. CNN provided the best overall results for NSL-KDD recording 93.56% accuracy, 94.61% precision, and 89.99% recall value. The false positive rate was only 0.019, and the *F*1-score was 0.91. The overall training time recorded was also better among all the deep learning models. Reported accuracy of CNN on NB15 and CIC-17 datasets was, however, lower (~85–86%). Considering a relatively larger sample size of NB15 and CIC-17 traffic and implying a greater presence of temporal characteristics would require RNN structures for improved classification in attack analysis. We, therefore, observe that for the hybrid (CNN + GRU) model, the tested accuracy for both NB15 and CIC-17 surpasses (>90%) the recorded value for NSL-KDD. The CNN model operating on NSL-KDD offers best precision (94.61%) while the recall. Value (90.29%) is better for the hybrid structure. Precision and recall values for the hybrid model exceed others in the remaining two datasets, slightly fairing better than CNN structure. Similarly, the FPR and *F*-1 values for the hybrid model (CNN + GRU) overall exceed other network structures with minor improvements over CNN.

LSTM and GRU structures both belong to RNN; however, their operational capability is different. The temporal characteristics of encrypted malicious traffic in the present work are also extracted independently using LSTM and GRU models. The LSTM training was completed over a longer time, and the reported performance was also relatively lower. One of the reasons for lower accuracy is the overfitting. Reduction in relevant parameters did not result in a significant improvement in the performance of LSTM to classify the encrypted traffic across the three datasets. Another reason for lower effectiveness of LSTM in classification can be attributed to the fact that the time-related characteristics of datasets such as NSL-KDD are not quickly obvious, and hence, LSTM may not be suitable.

The GRU network converged faster than LSTM, but the training time is longer, while the classification accuracy is better. Some of the GRU results partially exceeded even the CNN model, particularly accuracy (GRU-NB15 85.74%, CNN-NB15 85.14%) and *F*-1 score (GRU-NB15 93.21%, CNN-NB15 92.8%). For the shallow learning model, training accuracy was higher, but validation/testing reported very low effectiveness. Despite giving an appearance of high fitting on training data, the lack of a deep network structure made the generalization ability of the model far less than deep neural networks in real-time encrypted attack classification problem. Generally, the hybrid model outperformed for the different datasets in the experiment among the different models with minor exceptions reported against CNN. The individual performance of LSTM and GRU was relatively lower than CNN. The shallow neural network presented good training results; however, the generalization capability of the model was minimal.

The CNN + GRU model proves highly effective in comparison with the results of different models. CNN is capable of quick extraction of feature through convolutional operation. The small(er) kernel can extract detailed features of encrypted attack traffic and provide a good weight representation. The subsequent extraction of compressed features in encrypted traffic can be reduced using the flattening layer.

5.4. Hybrid Approach. The overall difference in classification performance between CNN and GRU models is minimal. The GRU model has an added advantage of extracting time-domain features, which may be quite useful in understanding and classifying the encrypted attack traffic using temporal characteristics. However, the GRU will only be of benefit if time-dependent features are present in the data. In the absence of time-domain features in input samples, the CNN offers higher accuracy and GRU will not be of much use. To build a model that has high applicability in either scenario, we propose a scheme based on the CNN + GRU hybrid test model. The CNN aids in quick extraction of feature whereas GRU helps with the time dimensionality of data if available. The advantages of CNN and GRU are, therefore, combined in the hybrid model.

The overall training time of CNN + GRU is relatively shorter as compared to GRU alone, and the model efficiency, however, is improved. The GRU component can retain the time features, and quick extraction of the CNN is also applicable. A pooling layer is not needed to prevent time characteristics filtering. CNN and GRU present a high fit for training data while maintaining classification accuracy. The complete hybrid model is illustrated in Figure 6.

The hybrid model can quite effectively classify encrypted attack traffic and reduce the data features dimensionality, a quite labor-intensive task in traditional machine learning approaches. Deep learning can automatically learn the important features and the anomalies in network behavior during training. The training phase in comparison with testing as recorded in the experiment can be time consuming and is directly proportional to the size of data. A greater amount of training data (if available), however, can also increase effectiveness of the deep learning model. The ROC (receiver operating characteristic) curve showing the performance of CNN and CNN + GRU for each of the datasets is presented in Figure 7.

It is quite notable that the difference between CNN and CNN + GRU hybrid classification models is rather small at all classification thresholds. The practical applicability of the

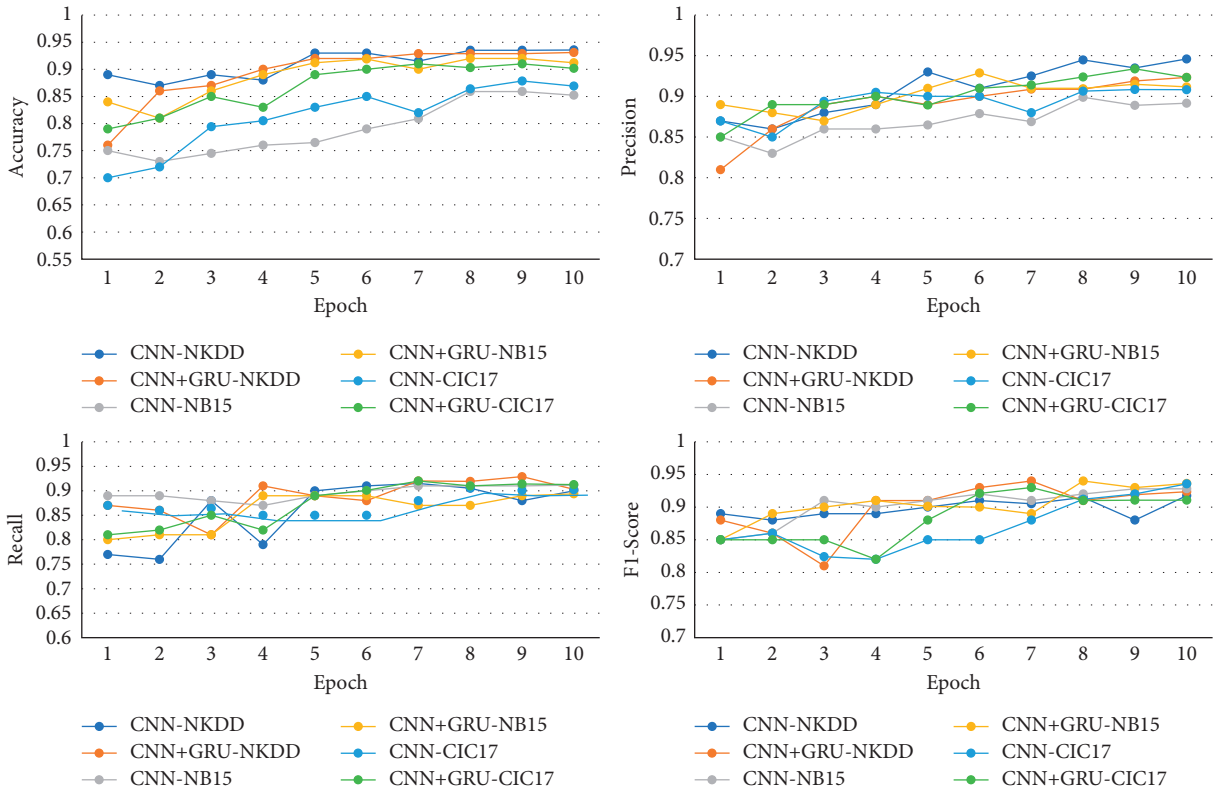

Figure 5: Performance comparison: CNN, LSTM, GRU, shallow, and hybrid models.

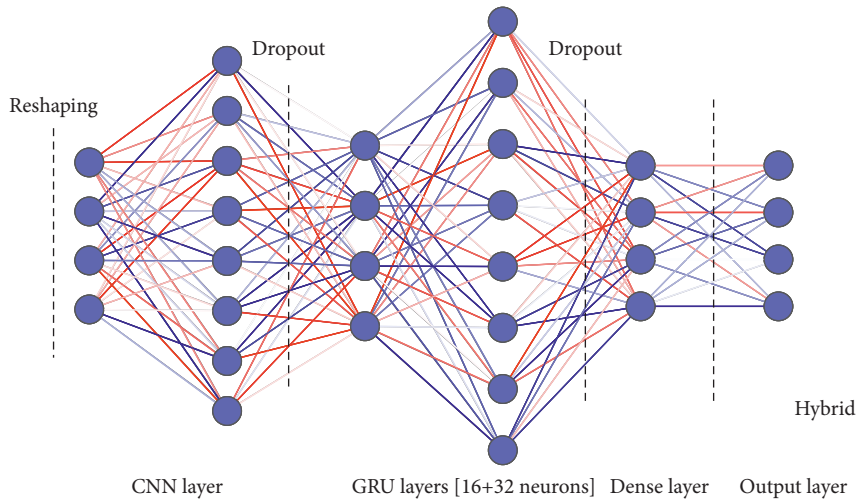

Figure 6: Structure of the optimal hybrid model.

hybrid approach for encrypted attack traffic classification would although be higher. The completeness in feature extraction ability of the CNN + GRU model would be useful for network managers aiming to classify the attack traffic in different types of small and large environments. Considering the prevalent networks in production, the CNN + GRU can be preferred over CNN, due to the inclusion of time-related features, and not eliminating any of the possible features that may prove useful in recognizing attack traffic.

The derived hybrid can be used with intrusion detection and prevention systems, firewalls, and any other access control machines to detect encrypted malicious traffic that cannot be discriminated using general purpose systems. A combination of the proposed hybrid model with a typical IDS is presented in Figure 8. The hybrid model has the ability to understand the structural features as well as time-related characteristics from network logging systems.

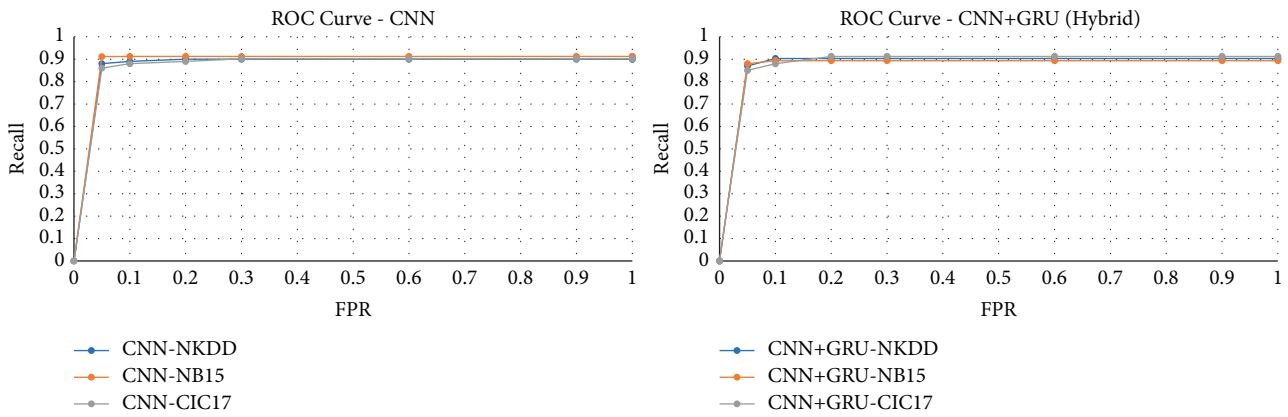

Figure 7: ROC-CNN and CNN + GRU (hybrid).

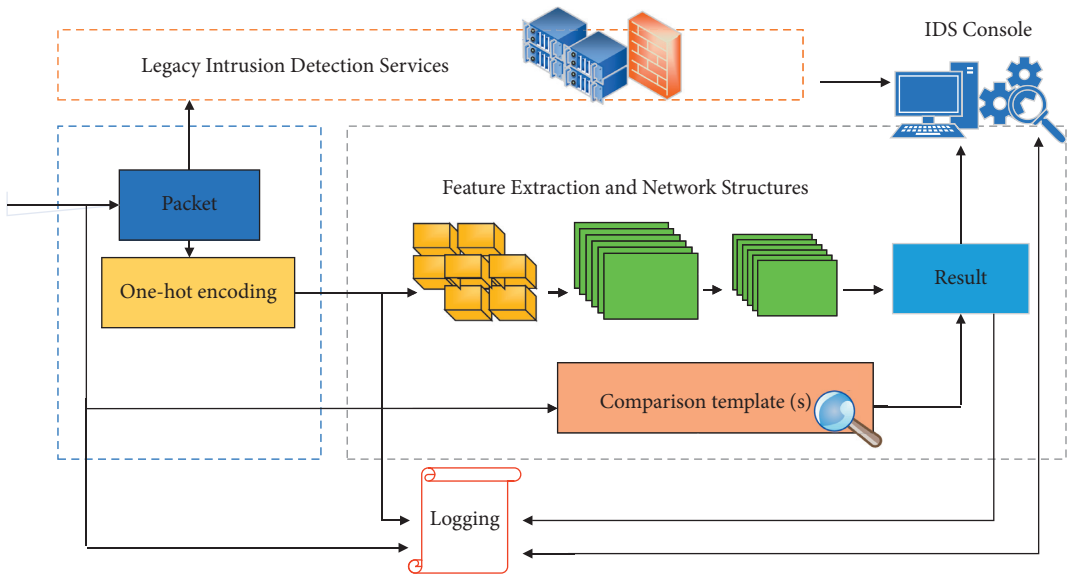

Figure 8: IDS implementation overview, the hybrid model.

5.5. Advantages and Limitations. The proposed hybrid model has certain notable advantages. The hybrid approach allows for automated feature extraction reducing manually intensive interventions and improve the overall efficacy of the classification. Encrypted malicious traffic can be detected in different network environments, and provided training data are available. The model is applicable to varying network systems as it is able to recognize the attack traffic using temporal as well as correlative features. The model uses parameters selected and optimized during the training and can deal with noise and outliers. The primary limitation is the lower efficacy (although relatively minute) in comparison with the CNN model where data or network traffic being monitored offers minimum or altogether lack of (any) temporal features.

6. Conclusion and Future Work

The present work proposes a hybrid approach using deep learning neural networks to detect and classify encrypted malicious network traffic. Deep learning is applied to learn as well as automate the reduction of feature set dimensions and discriminate between normal and encrypted attack traffic. Different deep learning network structures are tested on three different datasets, the NSL-KDD, UNSW-NB15, and CIC-IDS-2017 offering a mix of network traffic to allow realistic evaluation of the proposed approach. The combination of convolutional (CNN) and gated recurrent unit (GRU) neural models results in quick feature extraction and learning. This hybrid model fits well on network traffic through optimal feature adjustment. The convolutional network can learn the representative characteristics of network traffic in a relatively short time compared to other models. The gated recurrent unit network can determine and classify based on the time dimension of network traffic. Together, the hybrid CNN + GRU model provides high accuracy and can be employed in different network environments to uniquely classify malicious traffic.

Our future work will focus on further testing of hybrid structures on publicly available datasets to comprehensively evaluate the viability of deep learning models in encrypted

traffic classification. Additionally, we aim to explore multimodalities in traffic, as well as multitasking in a federated learning context that may allow greater insight into traffic monitoring while preserving network (data) privacy.

Data Availability

The data used to support the findings of this study are as follows: the NSL-KDD dataset, Canadian Institute of Cyber Security, website: https://www.unb.ca/cic/datasets/nsl.html. The UNSW-NB15 dataset, website: https://research.unsw.edu.au/projects/unsw-nb15-dataset. The CIC-IDS-2017 dataset, website https://www.unb.ca/cic/datasets/ids-2017.html.

Conflicts of Interest

The authors declare no conflicts of interest regarding the publication of this research paper.

References

[1] W. Di, F. Binxing, C. Xiang, and L. Qixu, "Botcatcher: *botnetdetection* system based on deep learning," *Journal on Communications*, vol. 39, no. 8, 2018.

[2] U. N. Dulhare and S. Rasool, "IoT evolution and security challenges in cyber space," *Countering Cyber Attacks and Preserving the Integrity and Availability of Critical Systems*, IGI Global, PA, USA, 2019.

[3] L. Guo, Q. Wu, S. Liu, M. Duan, and J. Sun, "Deep learning-based real-time vpn encrypted traffic identification methods," *Journal of Real Time Image Processing*, vol. 17, pp. 103–114, 2020.

[4] G. S. R. Hinton, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, 2006.

[5] H. Z. W. Yong, "Network traffic classification method based on deep convolutional neural network," *Journal of Communications*, vol. 39, no. 1, pp. 14–23, 2018.

[6] M. L. Martin, B. Carro, A.S. Esguevillas, and J. Lloret, "Network traffic classifier with convolutional and recurrent neural networks for Internet of Things," *IEEE Access*, vol. 5, Article ID 18042, 2017.

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

[8] NSL KDD Data Set, "Canadian Institute of Cyber Security," https://www.unb.ca/cic/datasets/nsl.html.

[9] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of the Military Communications and Information Systems Conference (MilCIS)*, November 2015.

[10] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, Funchal - Madeira, Portugal, January 2018.

[11] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges," *IEEE Transactions on Network and Service Management*, vol. 16, no. 2, pp. 445–458, June 2019.

[12] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapè, "MIMETIC: mobile encrypted traffic classification using multimodal deep learning," *Computer Networks*, vol. 165, Article ID 106944, 2019.

[13] C. Liu, L. He, G. Xiong, Z. Cao, and Z. Li, "FS-net: a flow sequence network for encrypted traffic classification," in *Proceedings of the IEEE INFOCOM 2019 - IEEE Conference on Computer Communications*, pp. 1171–1179, IEEE, Paris, France, April 2019.

[14] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[15] F. Rosenblatt, "The perceptron: a probabilistic model for infor-mation storage and organization in the brain," *Science*, vol. 65, no. 6, 1988.

[16] R. G. Radu, *The Monopoly of Violence in the Cyber Space: Challenges of Cyber Security*, Springer, Berlin Heidelberg, Germany, 2012.

[17] S. Sarbu, *The Cyber Threat and the Problem of Information security.a Critical Analysis of the Concepts of Cyber-Power and Cyber-Space*, Central And Eastern European Online Library, Frankfurt, Germany.

[18] W. U. Wei, "The research of cyberspace and communication network security problems," *Journal of China Academy of Electronicsand Information Technology*, vol. 6, no. 5, pp. 473–476, 2011.

[19] W. U. Wei, "Research on cyberspace and communication network security problems," *Radio and Communications Technology*, vol. 38, no. 3, pp. 1–4, 2011.

[20] V. Yosifova, R. Trifonov, A. Tasheva, and O. Nakov, "Trendsreview of the contemporary security problems in the cyberspace," in *Proceedings of the 9th Balkan Conference on Informatics*, Sofia, Bulgaria, September 2019.

[21] P. S. Labini, M. Cianfriglia, D. Perri et al., "On the anatomy of predictive models for accelerating GPU convolution kernels and beyond," *ACM Transactions on Architecture and Code Optimization*, vol. 18, 2021.

[22] B. Hong, G. Kim, S. Kim, JD. Kim, and B. Kim, "A study for accelerating of convolution operations based on multiple GPUs with MPI," in *Advances in Computer Science and Ubiquitous Computing. Lecture Notes in Electrical Engineering*, J. J. Park, S. J. Fong, Y. Pan, and Y. Sung, Eds., vol. vol 715Singapore, Springer, 2021.

[23] I. H. Sarker, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: an overview from machine learning perspective," *Journal of Big Data*, vol. 7, no. 1, 2020.

[24] Y. Gahi and I. E. Alaoui, "Machine Learning And Deep Learning Models For Big Data Issues," in *Machine Intelligence and Big Data Analytics for Cybersecurity Applications. Studies in Computational Intelligence*, Y. Maleh, M. Shojafar, M. Alazab, and Y. Baddi, Eds., vol. 919, Springer, Cham, Switzerland, 2021.

[25] X. Ma, Y. Niu, L. Gu et al., "Understanding adversarial attacks on deep learning based medical image analysis systems," *Pattern Recognition*, vol. 110, Article ID 107332, 2021.

[26] P. Dixit and S. Silakari, "Deep learning algorithms for cybersecurity applications: a technological and status review," *Computer Science Review*, vol. 39, Article ID 100317, 2021.

[27] S. Y. Kim, S. W. Yun, E. Y. Lee, S. H. Bae, and I. G. Lee, "Fast packet inspection for end-to-end encryption," *Electronics*, vol. 9, no. 11, 2020.

[28] J. Ning et al., "Pine: enabling privacy-preserving deep packet inspection on TLS with rule-hiding and fast connection establishment," in *Computer Security – ESORICS 2020. ESORICS 2020*, L. Chen, N. Li, K. Liang, and S. Schneider, Eds., Springer, Cham, Switzerland, 2020.

[29] I. Lancope, *Cisco to Showcase Lancope's Stealthwatch for Cyber Threat Defense at cisco Live*, Lancope Inc Cisco Live, Atlanta, Georgia, 2012.

[30] S. Gamage and J. Samarabandu, "Deep learning methods in network intrusion detection: a survey and an objective comparison," *Journal of Network and Computer Applications*, vol. 169, 2020 ISSN 1084-8045, Article ID 102767.

[31] H. Alaeddine and M. Jihene, "Deep network in network," *Neural Comput & Applic*, vol. 33, 2020.

[32] Y. Zhang, Y. Dong, C. Liu, K. Lei, and H. Sun, "Situation, trends and prospects of deep learning applied to cyberspace security," *Journal of Computer Research and Development*, vol. 55, no. 6, pp. 1117–1142, 2018.

[33] S. Rezaei and X. Liu, "Deep learning for encrypted traffic classification: an overview," *IEEE Communications Magazine*, vol. 57, no. 5, pp. 76–81, 2019.

[34] Z. Zou, J. Ge, H. Zheng, Y. Wu, and Z. Yao, "Encrypted traffic classification with a convolutional long short-term memory neural network," in *Proceedings of the 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS)*, Exeter, UK, June 2018.

[35] 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, Article ID 45182, 2019.

[36] M. Lotfollahi, R. S. H Zade, S. J. Mahdi, and M. Saberian, "Deep packet: a novel approach for encrypted traffic classification using deep learning," *SoftComputing*, vol. 24, 2017.

[37] G. Aceto, D. Ciuonzo, A. Montieri, and A. Pescapé, "DISTILLER: encrypted traffic classification via multimodal multitask deep learning," *Journal of Network and Computer Applications*, vol. 183–184, Article ID 102985, 2021.

[38] H. Huang, H. Deng, J. Chen, L. Han, and W. Wang, "Automatic multi-task learning system for abnormal network traffic detection," *International Journal of Emerging Technologies in Learning*, vol. 13, no. 4, pp. 4–20, 2018.

[39] Y. Zhao and J. Chen, D. Wu and J. Teng, S. Yu, Multi-task network anomaly detection using federated learning acm," in *Proceedings of the 10th International Symposium on Information and Communication Technology (SoICT)*, pp. 273–279, Hanoi, Vietnam, December 2019.

[40] G. Draper, A. H. Lashkari, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of encrypted and vpn traffic using time-related," in *Proceedings of The 2nd International Conference on Information Systems Security and Privacy (ICISSP)*, pp. 407–414, Rome, Italy, February 2016.

[41] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Computers & Security*, vol. 31, no. 3, pp. 357–374, 2012.

[42] L. Chang, C. Zigang, X. Gang, G. Gaopeng, Y. S. Ming, and H. T. Long, "Mampf: encrypted traffic classiffication based on multi-attribute Markov probability fingerprints," *Quality of Service (IWQoS) 2018 IEEEIACM 24th International Symposium on*, Banff, AB, Canada, June 2018.

[43] S. Ruder, "An overview of multi-task learning in deep neural networks," 2017.

[44] "Keras API reference/losses," https://keras.io/api/losses/.

[45] S. García, M. Grill, J. Stiborek, and A. Zunino, "An empirical comparison of botnet detection methods," *Computers & Security*, Elsevier, vol. 45, pp. 100–123, 2014.

[46] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal, "Chapter 10 - Deep learning," in *Data Mining*, H. Ian, E. Frank, A. H. Mark, and J. P. Christopher, Eds., pp. 417–466, Morgan Kaufmann, Fourth edition, 2017.

[47] P. Ramachandran, Z. Barret, and V. Quoc, "Searching for activation functions," 2017, https://arxiv.org/abs/1710.05941.

[48] I. Goodfellow, Y. Bengio, and A. Courville, "6.2.2.3 softmax units for multinoulli output distributions," *in Deep Learning*, pp. 180–184, MIT Press, Cambridge, MA, USA, 2016.

[49] M. Tavallaee, E. Bagheri, W. Lu, and A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Submitted to Second IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, Ottawa, ON, Canada, July 2009.

[50] N. Moustafa, G. Creech, and J. Slay, "Big Data Analytics for Intrusion Detection System: Statistical Decision-Making Using Finite Dirichlet Mixture Models," in *Data Analytics and Decision Support for Cybersecurity*, I. P. Carrascosa, H. K. Kalutarage, and Y. Huang, Eds., Springer, Berlin, Germany, 2017.

[51] A. I. Federated, "Ecosystem," https://github.com/FederatedAI/FATE.

[52] "Keras 2.4.0, python-based deep learning API," https://github.com/keras-team/keras/releases.

[53] "Tensorflow Open Source Machine Learning Platform," https://github.com/tensorflow/tensorflow.

[54] T. Dozat, "Incorporating nesterov momentum into adam," *ICLR Workshop*, 1, San Juan, Puerto Rico, May 2016.

[55] 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, Article ID 102890, 2021.

[56] J. Bradbury, S. Merity, C. Xiong, and R. Socher, "Quasi-recurrent neural networks," 2016, https://arxiv.org/abs/1611.01576.