Contents lists available at ScienceDirect

Digital Communications and Networks

journal homepage: www.keaipublishing.com/dcan

IDS-INT: Intrusion detection system using transformer-based transfer learning for imbalanced network traffic

Farhan Ullah[a], Shamsher Ullah[b], Gautam Srivastava[c,d,f,*], Jerry Chun-Wei Lin[e]

[a] School of Software, Northwestern Polytechnical University, Xian, 710072, Shaanxi, China
[b] Knowledge Unit of Systems and Technology, University of Management and Technology, Sialkot, 51040, Pakistan
[c] Department of Math and Computer Science, Brandon University, Brandon, MB R7A 6A9, Canada
[d] Department of Computer Science and Math, Lebanese American University, Beirut, 1102, Lebanon
[e] Western Norway University of Applied Sciences, Bergen, Norway
[f] Research Centre for Interneural Computing, China Medical University, Taichung, Taiwan, China

A R T I C L E I N F O

Keywords:
Network intrusion detection
Transfer learning
Features extraction
Imbalance data
Explainable AI
Cybersecurity

A B S T R A C T

A network intrusion detection system is critical for cyber security against illegitimate attacks. In terms of feature perspectives, network traffic may include a variety of elements such as attack reference, attack type, a subcategory of attack, host information, malicious scripts, etc. In terms of network perspectives, network traffic may contain an imbalanced number of harmful attacks when compared to normal traffic. It is challenging to identify a specific attack due to complex features and data imbalance issues. To address these issues, this paper proposes an Intrusion Detection System using transformer-based transfer learning for Imbalanced Network Traffic (IDS-INT). IDS-INT uses transformer-based transfer learning to learn feature interactions in both network feature representation and imbalanced data. First, detailed information about each type of attack is gathered from network interaction descriptions, which include network nodes, attack type, reference, host information, etc. Second, the transformer-based transfer learning approach is developed to learn detailed feature representation using their semantic anchors. Third, the Synthetic Minority Oversampling Technique (SMOTE) is implemented to balance abnormal traffic and detect minority attacks. Fourth, the Convolution Neural Network (CNN) model is designed to extract deep features from the balanced network traffic. Finally, the hybrid approach of the CNN-Long Short-Term Memory (CNN-LSTM) model is developed to detect different types of attacks from the deep features. Detailed experiments are conducted to test the proposed approach using three standard datasets, i.e., UNSW-NB15, CIC-IDS2017, and NSL-KDD. An explainable AI approach is implemented to interpret the proposed method and develop a trustable model.

1. Introduction

Currently, the widespread use of the Internet has made life more convenient for everyone, but it has also made cybersecurity threats more difficult to combat. With the advent of Internet of Things (IoT) devices such as smart objects, smart handheld devices, and smart sensing technologies, it is now possible to process massive amounts of data quickly and efficiently. Thus, such systems are consequently vulnerable to a wide range of malicious attacks and security flaws [1]. The first line of defense in a computer network is Intrusion Detection Systems (IDS). IDS are divided into two primary categories: host-based IDS (H-IDS), and network-based IDS (N-IDS) [2]. The H-IDS is installed on a host computer, whereas the N-IDS is distributed throughout a network system. Furthermore, both types of IDS use signature-based and anomaly-based detection methods in their operations. Phishing violations are the most common type of cyberattack, followed by Distributed Denial of Service (DDoS) attacks and security breaches. Additionally, ransomware attacks are frequent and their variants are growing [3]. In recent years, DDoS attacks have become more prevalent, and their planning and objectives have come into the spotlight. Advanced Persistent Threat (APT) attacks have slowly spread to important industries and become more common during big events and times when security is weak. If an organization or a person is attacked by a network intrusion, they can suffer significant losses. So, the threat of network intrusion has

* Corresponding author. Department of Math and Computer Science, Brandon University, Brandon, MB R7A 6A9, Canada.
E-mail addresses: farhan@nwpu.edu.cn (F. Ullah), shamsherullah@nwpu.edu.cn (S. Ullah), srivastavag@brandonu.ca (G. Srivastava), jerrylin@ieee.org (J.C.-W. Lin).

https://doi.org/10.1016/j.dcan.2023.03.008
Received 23 October 2022; Received in revised form 31 January 2023; Accepted 14 March 2023
Available online 17 March 2023

grown into a big problem that needs to be fixed right away. Numerous researchers are continuously working to develop a secure and reliable network intrusion detection system to achieve this goal [4].

Traditional network intrusion detection systems can be divided into misuse detection and anomaly detection [5]. Misuse detection uses regulatory matching or feature matching to determine if the data is an intrusion, whereas anomaly detection determines if data is abnormal. However, both approaches have drawbacks, including low detection rates and high false alarm rates. Researchers are using machine learning and deep neural models to improve N-IDS results with intelligent systems [6]. In the early days of N-IDS, traditional machine learning was usually used to look into how a single model's prediction could be made. However, the prediction accuracy obtained from a single model is quite limited. Models such as k-nearest neighbor [7], naive bays [8], decision tree [9], support vector machine, random forest [10], and other algorithms have been used in N-IDS. Although these techniques are simple and require little training, their accuracy is low. When compared to N-IDS built using conventional machine learning methods, N-IDS built using deep learning achieves superior accuracy. For instance, N-IDS have used Multi-Layer Perceptron (MLP) [11], Convolutional Neural Network (CNN), Recurrent Neural Networks (RNN) [12], Long Short-Term Memory (LSTM), and other related algorithms to obtain detection accuracy. Even though these algorithms have made IDS more accurate, they are very hard to program and take a long time to train. Bidirectional Encoder Representations from Transformers (BERT) base [13] classify malware such as adware, ransomware, scareware, SMS, etc. There are fewer layers and parameters in this basic version of BERT. For instance, the BERT base is comprised of 12 layers of transformer blocks, 768 hidden blocks, 12 self-attention heads, and 110 M trainable parameters. Furthermore, this study classifies malware using traditional machine learning algorithms, such as Gaussian Naive Bayes (GNB), Support Vector Machine (SVM), Decision Tree (DT), etc. The objective of this study is to detect network-based intrusion, such as fuzzers, port scanners, backdoors, DoS, exploits, etc., using an advanced version of the BERT model named BERT large. We developed the BERT large model to extract train features from network traffic using in-depth analysis. This method has 24 encoded layers, 1024 hidden sizes, 16 self-attention heads, and 340 M parameters. To detect network-based intrusion, we used a combination of deep learning models, such as CNN-LSTM. Following that, extensive comparisons with other combined deep learning approaches, such as CNN-RNN and CNN-GRU, are made using three different standard datasets. The findings also support the claim that increasing the size of the model from BERT base to BERT large using deep learning methods leads to better results.

In the past, machine learning and deep learning-based N-IDS have focused on model performance rather than semantic analysis of the network data. Fig. 1 depicts the general N-IDS scenario in which the devices are connected and communicate in real time. N-IDS inspects network traffic and hosts to identify possible intrusions. This system is linked to a network node, like a hub or switch, so that network activity can be tracked. N-IDS surveillance points are placed in high-traffic areas of the network to inspect data packets for signs of bad behavior. Attackers may employ several malicious scripts to damage a target device. In terms of behavioral segmentation, semantic-based feature analysis can reveal potentially damaging scripts. Before feeding the network data into the deep learning model, it can be semantically analyzed to extract train features [14,15]. For semantic analysis, detailed information for each type of attack, such as host information, attack name, and attack reference, can be combined with other features. Overall, each security feature only gives a limited view of the security threat and often only looks at one type of attack. Furthermore, the ability of various features to differentiate between abnormal activities varies considerably. Because of this, most existing IDS were made to collect a full picture by using a mix of features instead of just one type. Furthermore, attackers leverage existing feature-related experiences to create intrusion variants to avoid detection. It is desirable to develop novel feature sets to supplement existing knowledge and expand the feature combination space. This makes it more difficult for intruders to avoid detection.

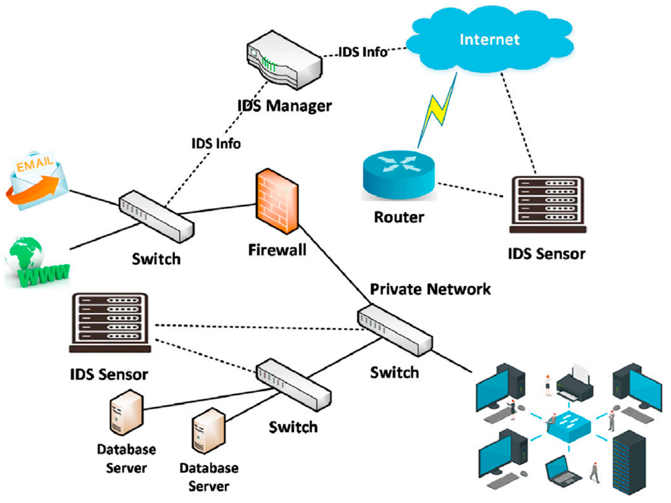


Fig. 1. Network-based intrusion detection system (N-IDS).

This paper presents a novel IDS-INT approach that uses semantic-based analysis to detect various types of attacks. The multi-head attention-based transformer method is designed to extract train features for accurately detecting various types of attacks. The main contributions of the paper are the following:

1. Multi-head attention-based transformer approach is designed using the BERT large model. This makes it possible to mine network data and pull out train features, which helps wireless devices work better.
2. Because normal traffic samples surpass minority attacks, the deep learning model may be biased. The SMOTE method is used to balance each class to solve the model's bias problem.
3. To classify various types of network intrusion attacks, a combined approach of CNN and LSTM deep learning models is developed. The effectiveness of the proposed approach is validated by comparing it with other cutting-edge deep learning methods.
4. An explainable AI experiment is conducted to interpret the designed approach. This assists in analyzing the most reliable and efficient features that contribute to a specific type of attack.

The remaining part of the paper is organized as follows: Section 2 discusses the related and recently published works, Section 3 briefly explains the proposed IDS-INT scheme, the experimental analysis is represented in Section 4 and finally, Section 5 concludes the paper.

2. Related work

Network intrusion detection can be viewed as a supervised anomaly detection problem, which is also a classification problem. Recently, several studies have been proposed to obtain unique patterns from attack intrusions and differentiate attacks from normal traffic, ranging from traditional statistical techniques to deep learning-based methods.

2.1. Semantic-based methods

Enterprise systems frequently generate a substantial volume of logs to track runtime status and activities network traffic data. Traditionally, intrusion detection requires the parsing of unstructured network data by analyzing Packet Capture (PCAP) files. Such data may contain useful information about the specific attack, such as the attack name, reference, host information, and so on. Semantic tools can be used to pre-process network data to extract meaningful information and remove noisy data. It may also aid in reducing the load on the N-IDS [16]. Seyyar et al.

[14] proposed a model for a web IDS that can distinguish between normal and abnormal URLs. During the URL analysis phase, the method employs the BERT transformer model and then CNN to classify various types of attacks. The experimental results showed an accuracy of more than 96%. To address the semantic gap, li et al. [17] used the word2vec model and implemented the TF-IDF weighted mapping of HTTP traffic to generate a low-dimensional features vector. Using the HTTP CSIC 2010, UNSW-NB15, and malicious-URL datasets, experimental results show high detection accuracy, a high true positive rate, and a low false positive rate. Huang et al. [18] proposed the HitAnomaly method which uses a BERT to construct log template segments and attribute values for anomaly detection. Log sequence and parameter settings-based encoders are designed to extract the sequenced data. Compared to log-based anomaly detection methods, the proposed method performs better. Min et al. [19] suggested an intrusion detection system that uses statistical and payload features. Word semantics and text-CNN are used to gather useful features from payloads. The proposed method provides the best detection accuracy. The proposed study provides the best classification results for each type of attack, with more than 98% accuracy.

2.2. *Features selection methods*

The method of feature selection can also be used as a pre-processing step in machine learning to reduce computation costs. It aims to remove unnecessary features while maintaining or even improving IDS performance. Aslahi-Shahri et al. [20] used a combination of a support vector machine and a genetic algorithm to decrease the number of features in the KDD CUP-99 dataset from 41 to 10. The experiments demonstrate that the proposed method achieves better true positive rates while maintaining very low false positive rates. To enhance the process of selecting features for N-IDS, Alazzam et al. [21] used a Pigeon Inspired Optimizer (PIO). The suggested method lowered the number of features in the KDD CUP99, NSL-KDD, and UNSW-NB15 datasets from 41 to 7, 5, and 5, respectively. Furthermore, it maintains high true positive rates, and accuracy, and decreases the amount of time to build a decision tree. To select the optimal subset for network IDSs, Khammassi et al. [22] utilized logistic regression as a selection method and as a trained model. Their method yields high detection accuracy with 18 features for KDD CUP-99 and 20 features for UNSW-NB15. Even though feature selection-based techniques can minimize feature dimension, their effectiveness is heavily dependent on the quality of features with semantic details. It cannot ensure the best results in every data situation, particularly in the case of imbalanced data. Furthermore, current feature selection strategies are based on heuristic rules and metrics, limiting their ability to learn complex relationships among features.

2.3. *Deep learning methods*

Deep learning has proven to be effective in a variety of fields, including image feature extraction, text mining, and tabular data. Deep learning methods are being used by researchers to automatically learn feature representation in N-IDS [6,23]. Vinayakumar et al. [24] presented a hybrid IDS using distributed deep learning and Deep Neural Networks (DNN). It employs five hidden layers to handle and analyze large amounts of data in real time. They determine the optimal number of layers and nodes by running the proposed method on six datasets for binary and multiclass classification, including KDD CUP99, NSL-KDD, WSN-DS, and UNSW-NB15. Jiang et al. [25] developed a multi-channel IDS using long short-term memory recurrent neural networks (LSTM-RNN). It reveals attack behavior as sequential data by mining the information of a specific attack. Autoencoders are also extensively used in N-IDS to discover the reflection of network behavior, which can then be used to detect abnormal traffic. Shone et al. [6] used Random Forest (RF) and non-symmetric deep autoencoder. The method employs stacked autoencoders, each of which contains three hidden layers; the output of the auto encoder's final encoding layer serves as the input to the softmax classifier. In addition, several deep learning-based N-IDS use a combination of unsupervised Autoencoders (AE) and a softmax layer to obtain end-to-end training. Naseer et al. [26] looked into the use of DNN for anomaly detection. In the implementation, Deep Convolutional Neural Networks (DCNN), various autoencoders, and LSTM-RNN were used. Such DNN models were pitted against traditional machine learning techniques. The experimental results showed that DCNN achieved an 85% detection rate on the test data, which was better than standard machine learning methods.

A network is frequently compromised for one of the following three reasons: a) Cyber terrorism: Hacking and activism are combined to form cyber terrorism. The intruders wish to invade to prove a political objective or social cause. b) Stealing Money: The purpose of this intrusion is to steal money or information from the victim. Generally, the intent is to harm the victim for monetary benefit. c) Spying: An enemy or alliance is spied on by a state-sponsored network. The following are some of the significant losses caused by network intrusion if a company does not use IDS: a) Data corruption: A high volume of requests or unauthorized requests may taint the vital details about the company or its customers. Orders and operations may be altered, and client transactions may take longer to process. b) Economic Loss: Perks and bonuses may be required to gain the trust of clients and stakeholders. Potential transactions are lost if an attack occurs during the holidays or sales, resulting in additional economic losses. The cost of restoring the defective product is an additional expense. c) Data theft: For intruders, client data is a valuable resource. Social engineering and other techniques can be used to obtain their address, contact information, email addresses, and even billing information. d) Interrupted execution: The company may halt activities until it recovers from the attack, delaying normal service. e) Reputation risk: Loss of credibility can be disastrous for a company. Customers, adversaries, financial leverage, and market share effects would make it more difficult for the company to recover. Existing intrusion detection methodologies only use deep learning by reshaping categorical features into one-hot encoding, which is incapable of simulating complex intrusion behavior. To learn a strong picture of N-IDS, we combine the semantic information of attacks feature learning and implicit deep learning. Semantic-based and balanced features are used in N-IDS to address specific problems such as diverse features and limited labeled data in abnormal classes.

3. Proposed IDS-INT scheme

Fig. 2 shows the proposed research framework. The IDS manager analyzes and collects network data in the form of PCAP files. The network traffic data is recorded in the PCAP file in encrypted form. The network parser is used to convert the PCAP file to a human-readable format and extract flow events. These flow events could include malicious network behavior such as malicious scripts or malicious URLs. The transfer learning approach based on transformers is intended to extract semantic features from flow events. As normal network data may be greater than the abnormal data, the SMOTE method is then used to balance the features. The CNN model extracts deep features from balance features, and the CNN-LSTM model detects network-based intrusion.

3.1. *Network traffic analysis*

Packet capture is a networking technique that uses PCAP to intercept data packets as they travel through a network and store them for later analysis. This file contains network traffic information and is used to assess the underlying data exchange between botnets. By inspecting these packets, IT analysts can detect flaws and resolve network issues. They also aid in network traffic management and the detection of specific data traffic. The packets are encoded in the PCAP file, and we used the Wireshark tool to decode them into a human-readable format. The port number can explain network data transmission between communicating nodes. This number may differ if a node redirects traffic to a different

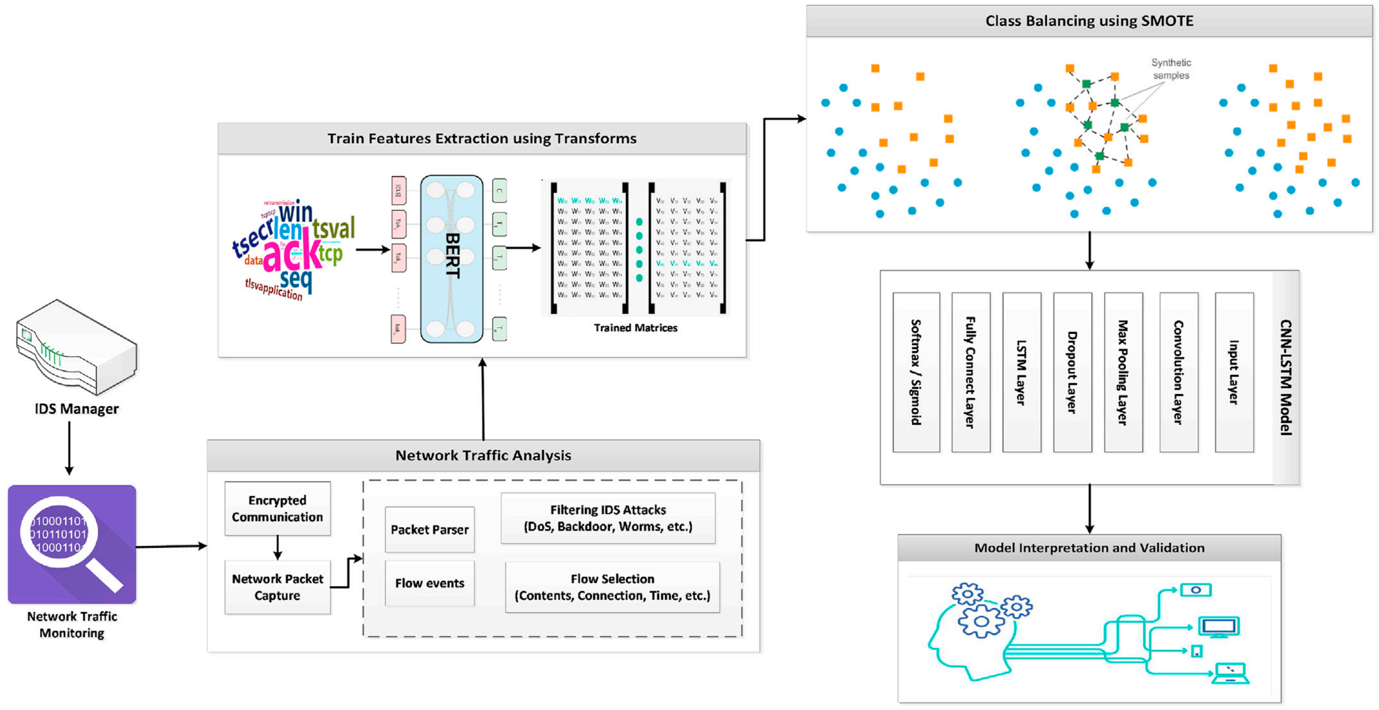


Fig. 2. Intrusion detection system using transformer-based transfer learning and explainable AI.

Table 1
Detailed information about different types of attacks filtered from PCAP.

Start time-Last time	Attack category-subcategory	Protocol	Source IP-Port	Destination IP-Port	Attack Name	Attacks Reference
0.000011	Reconnaissance-HTTP	tcp	175.45.176.0–13284	149.171.126.16–80	Domino Web Server Database Access:/doladmin.nsf [a]	– –
0.000008	Exploits-Unix ‘r’ Service	udp	175.45.176.3–21223	149.171.126.18–32780	Solaris rwalld Format String Vulnerability [b]	CVE 2002–0573 [c]
0.000004	Exploits-Browser	tcp	175.45.176.2–23357	149.171.126.16–80	Windows Metafile (WMF) SetAbortProc() Code Execution [009] [d]	CVE 2005–4560 [e]
0.000005	Exploits-Miscellaneous Batch	tcp	175.45.176.2–13792	149.171.126.16–5555	HP Data Protector Backup [f]	CVE 2011-1729[g]
0.000009	Exploits-Cisco IOS	tcp	175.45.176.2–26939	149.171.126.10–80	Cisco IOS HTTP Authentication Bypass Level 64[h]	CVE 2001–0537 [i]
0.000028	DoS-Miscellaneous	tcp	175.45.176.0–39500	149.171.126.15–80	Cisco DCP2100 SADownStartingFrequency Denial of Service [j]	http://www.exploit-db.com/exploits/21523/[k]

[a] (https://strikecenter.bpointsys.com/bps/strikes/recon/http/domino/access_domino_doladmin_nsf.xml).
[b] (https://strikecenter.bpointsys.com/bps/strikes/exploits/rservices/solaris_rwall_format_string.xml).
[c] (http://cve.mitre.org/cgi-bin/cvename.cgi?name=2002%2d0573)BID 4639 (http://www.securityfocus.com/bid/4639)CVSS-High (https://strikecenter.bpointsys.com/bps/reference/CVSS/).
[d] (https://strikecenter.bpointsys.com/bps/strikes/exploits/browser/wmf_009.xml) [e](http://cve.mitre.org/cgi-bin/cvename.cgi?name=2005%2d4560)BID 16074 (http://www.securityfocus.com/bid/16074)OSVDB 21987).
[e] (https://strikecenter.bpointsys.com/bps/strikes/exploits/misc/cve_2011_1729.xml).
[f] (http://cve.mitre.org/cgi-bin/cvename.cgi?name=2011%2d1729)BID 47638 (http://www.securityfocus.com/bid/47638)OSVDB 72188 (http://www.osvdb.org/72188)CVSS-Critical)-.
[h] (https://strikecenter.bpointsys.com/bps/strikes/exploits/ios/cisco_auth_bypass_level_64.xml).
[i] (http://cve.mitre.org/cgi-bin/cvename.cgi?name=2001%2d0537)BID 2936 (http://www.securityfocus.com/bid/2936)OSVDB 578 (http://www.osvdb.org/578)CVSS-High)/).
[j] (https://strikecenter.bpointsys.com/bps/strikes/denial/misc/cisco_dcp2100_denial_of_service.xml).
[k] (http%3a%2f%2fwww.exploit%2ddb.com%2fexploits%2f21523%2f)CVSS-High (https://strikecenter.bpointsys.com/bps/reference/CVSS/7.8).

port. To decrease the load on N-IDS, we categorized data according to the nature of features such as basic, contents, connection, time, etc. Further, each network flow contains detailed information such as type of attack, attack name, attack reference, host information, and malicious scripts that can be filtered from PCAP as shown in Table 1. We incorporate this data into the proposed IDS-INT to develop an intelligent transfer learning approach. However, such data is noisy and cannot carry meaningful semantics for the detection system. For instance, the attack name and attack reference for Reconnaissance and exploit attacks include messy and noisy information. Such data can be filtered to preserve actual semantics. We develop a semantic tokenizer that is capable of filtering such data. The following are the main steps in network traffic analysis:

- To prevent redundant data, eliminate features from input segments that are concurrently identical.
- Special symbols may not have enough information to differentiate relevant network traffic and are thus removed from data.
- Because distinct pattern lengths overwhelm neural network models, N-IDS must standardize sequence length.
- In IDS-INT, the lengths are balanced using a predetermined pattern length L. Patterns longer than L retain their first L names, while those that are lower than L are unified using zero-padding.

3.2. *Transformer-based transfer learning*

BERT is an open-source framework for machine learning and Natural Language Processing (NLP). It is a pre-trained algorithm for comprehending the interpretation of lengthy and complex texts. Transformers is a deep learning model that is used by BERT. In this model, each input and output are connected, and the attention head between them is dynamically computed [13,27,28]. It is designed to help devices understand the meaning of an ambiguous text by establishing context using nearby text. Non-contextual models can only give a word description of a term, no matter how it is used in the network flow. It is a contextual model that gives different meanings to words in a sentence based on how they relate to other words in the sentence. We used the transformer-based transfer learning method with the Bert large model to pull train features out of network traffic. Fig. 3 depicts the transformer architecture using the encoder and decoder.

Network features are encoded and decoded using embedding and positional encoding layers. The embedding layer records the definition of network flows. The transformer has two embedding layers. The sequences of network features are provided as input to the input embedding. The target sequence is supplied to the second embedding layer after it has been moved one position to the right and a start token is placed in the first place. The embedding layer converts network features into embedding vectors to better represent network attacks. The position encoding is computed individually from the features sequential manner. Features sequence matrices are used by the embedding and position encoding layers. The encoding size for position encoding is the same as the embedding size. As a result, it generates a similar-shaped matrix that can be integrated into the embedding matrix. The encoder stack has multi-head attention and feed-forward layers for each encoder. A feed-forward layer and two multi-head attention layers are included in each decoder. The output layer can be linear or softmax depends whether the features are binary or multi-class classification.

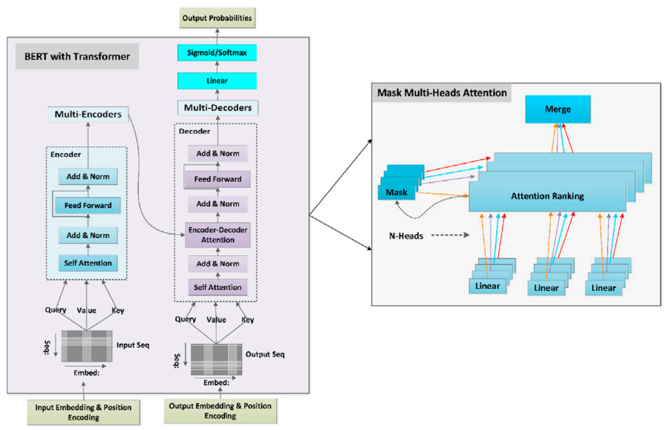


Fig. 3. Semantic features processing using transformers.

3.3. *Handling imbalance network traffic*

The nature of network flows allows for the growth of a class imbalance problem. Normal network flows always surpass abnormal network traffic because the intruder may attack the network at a specific time and date. Furthermore, they may use various types of attacks to compromise network behavior. The network dataset contains a diverse set of features related to various types of attacks, and each network type corresponds to a class label. When one class dominates another, it can be difficult to train the deep learning model for each class effectively. It has a significant impact on the proposed IDS-INT's evaluation criteria and detection accuracy. During training, the deep learning model may concentrate on the major class while ignoring the minor. This can result in high accuracy for the major class but low accuracy for the minor class as shown in Fig. 4.

To address this issue, we oversampled minority classes using the Synthetic Minority Over-Sampling (SMOTE) technique. It introduces various minority class samples based on their resemblance to the previous samples [29]. This brings the influence of minor classes closer to that of major ones. The following is how SMOTE operates:

- The Euclidean distance method is used to find the k-nearest neighbor for each minor class $x_i \in S_{\min}$.
- Next, select the random and nearest instance x_j in a cluster of nearest neighbors of x_i. Equation (1) is now used to generate a new sample.

$$x_{new} = x_i + |x_i + x_j| + \delta \tag{1}$$

The distribution of the newly generated samples is determined by a random factor, denoted by $\delta[0, 1]$.

3.4. *Deep features extraction*

The CNN network is developed to extract deep features by examining a large number of features, which helps the detection model run more efficiently. To accomplish this, balanced network features are fed into the CNN model. Several researchers [30–32] have used the CNN model for IDS. The CNN model outperforms other models when dealing with text, images, and video files. We employ a 1-D CNN network that consists of convolutional, max-pooling, dropout, and fully connected layers. The network features are recurrently cycled along by convolution to generate the most accurate network representations. A feature map, which is created by each filter, contains a clean collection of characteristics. The hyper-parameters are adjusted to find the appropriate number of filters. Four convolution layers are used, each with a different number of filters (64, 128, 156, and 512). Using max-pooling layers reduces the size of the feature set, number of features, and the amount of work that needs to be done to process the data. This layer also makes a feature map of the most important things from the previous set. We also integrate deep CNN with the Keras batch normalization layer. The final output mean and standard deviation are kept close to zero and one, respectively, thanks to the batch normalization method. This helps to steady the learning process and shorten the time it takes for deep networks to become fully trained. The proposed CNN network has softmax and dropout layers to deal with overfitting. The output of the CNN network is shown in Equation (2)

$$o_k^1 = f\left(c_k^1 + \sum_{i=1}^{N_{l-1}} ConID(X_{ik}^{l-1}, t_i^{l-1})\right) \tag{2}$$

where c_k^1 is the parameter bias of the kth neuron in the first layer, t_i^{l-1} is the outcome of the ith neuron in layer $l-1$, X_{ik}^{l-1} is the kernel strength from the ith neuron in layer *l-1* to the k^th neurons in layer *l*, and "f()" is the activation function.

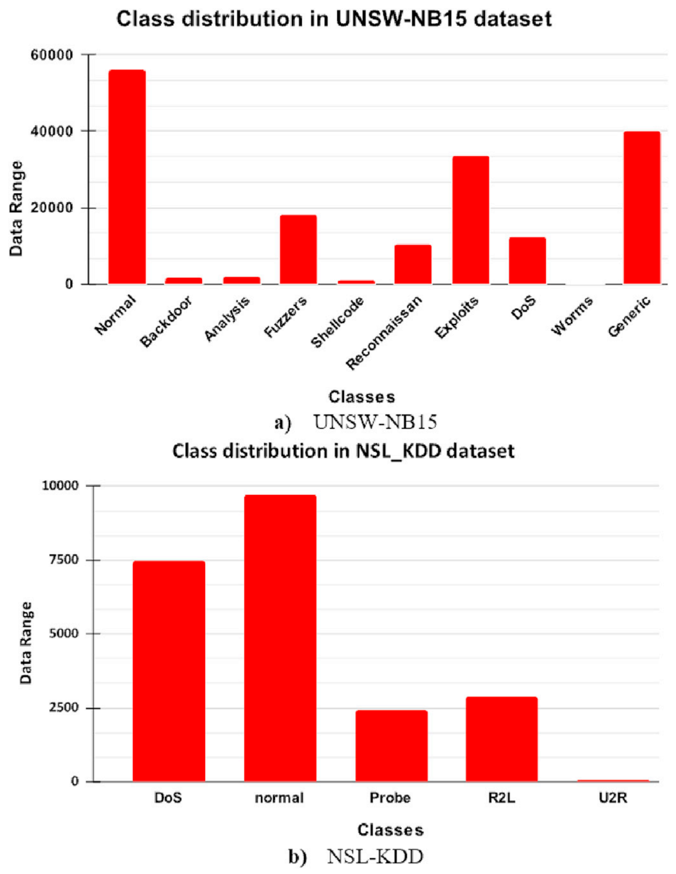


Fig. 4. Imbalance class distribution in UNSW-NB15 and NSL_KDD datasets.

3.5. Network-based intrusion detection system (N-IDS)

The proposed method introduces a CNN-LSTM [33,34] model for the automatic detection of network-based intrusion attacks to benefit from both models. CNN networks use max-pooling layers to extract features for the fully connected layer. The proposed CNN-LSTM model, on the other hand, sends deep features to the LSTM layer instead of the fully connected layer. The LSTM network is good at finding long-term and short-term correlations, while the CNN network is good at representing and understanding network feature vectors. The proposed CNN-LSTM model consists of two stages, as depicted in Fig. 5.

The convolution, dropout, and max-pooling layers are used in the first stage, while LSTM and dropout layers are used in the second stage. The convolution layers encode the network's feature set, and the LSTM layers decode it. Before information is sent to a fully connected layer in an intrusion detection system, it is flattened. The cell state and its various gates are central to LSTM. The cell state acts like a highway for data links, sending important information down the sequence alignment network. The LSTM model is made up of one storage unit and three interaction gates: the input, forget, and output gates. The memory module seems to be the most essential component of LSTM. Because this unit remembers each communicating link's current and previous states, To pursue data communication and information between current and previous states, the sigmoid activation function is used. The resultant values can be in the range between 0 and 1. The value near 0 indicates forgetting, while the value near 1 indicates keeping. The input gate decides how much information from the network should be kept in the unit state at a time *t.* The forget gate controls data flow to the input gate at the time *t-1*. The output gate regulates the parameter settings. Furthermore, this gate supervises the operation of each communicating link's hidden state. It is critical to handle the hidden state because it incorporates information from previous inputs. The previous hidden state is first integrated with the current input using a sigmoid function. The newly altered cell state is then passed on to the tanh function. The hidden state information can then be computed by multiplying the sigmoid by the tanh function. The resulting value is then used to predict the hidden state information. Following that, the information from the new cell state and hidden state is passed on to the next input sequence for detection. Equation (3) describes how the model works.

$$\begin{aligned}
i_t &= \sigma(V_{ixt} + W_i h_{(t-1)} + b_i)\\
f_t &= \sigma(V_{fxt} + W_f\ h_{(t-1)} + b_f)\\
\widehat{c_t} &= tanh(V_c\ X_t + W_c h_{(t-1)} + b_c)\\
c_t &= (f_t A C_t + W_c h_{(t-1)} + i_t A \widehat{c_t})\\
o_t &= \sigma(V_o x_t + W_o h_{(t-1)} + b_o)\\
h_t &= o_t A tanh(c_t)
\end{aligned} \tag{3}$$

The symbol x_t represents the input at the time *t* the symbols *x* and *w* represent the weight matrices, and the symbols *b* and *h* represent the bias and hidden states, respectively. The symbols *σ* and *tanh* represent the activation functions. The letters i_t, f_t, o_t, and c_t, represents the input gate, forget gate, output gate, and memory cell, respectively. Algorithm 1 depicts the entire procedure for the proposed method.

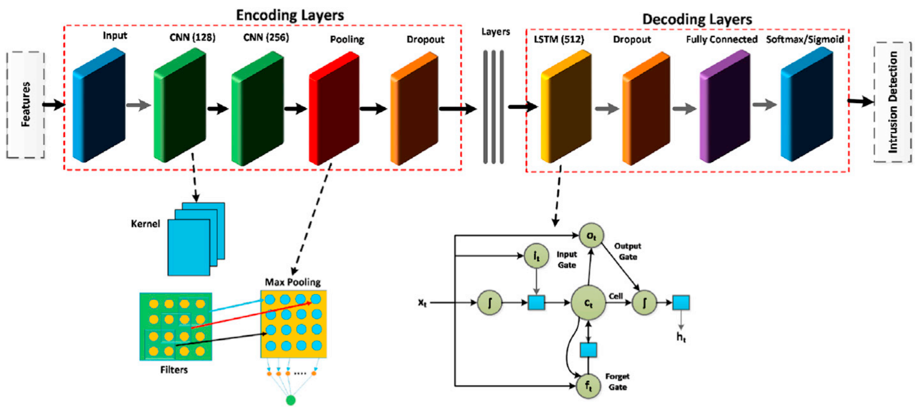


Fig. 5. CNN-LSTM architecture for IDS-INT.

```
Algorithm 1: IDS-INT for imbalanced network traffic
   Input: PCAP
   Output: D2AT // where D2AT stands for Detect
           Different Type of Attacks
   initialization;
1  Filtering ($Filter_{Info}$) ← $PP(PCAP)$ // where PP
     stands for Packet Parser ;
   if $Filter_{Info}$ is $Filtered$ then
2      $Filtered$ ← $Attack_{Categary} == A_C$;
3      $Filtered$ ← $Attack_{Reference} == A_R$;
4      $Filtered$ ← $Host_{Info} == H_I$;
5      $Filtered$ ← $Malicious_{Script} == M_S$;
       else ;
       if $A_C == A_C, A_R == A_R, H_I == H_I, M_S == M_S$
        then
           Filtered the following:
6          Categorize the attacks;
7          Referenced the attacks;
8          Hosted the Information;
9          Malicious Scripts ;
       else if Apply Flow selection then
10         ($Filtered_{Info}$) → $Generate_{Subsets}$ ;
             /* where $G_S$ stands for
             $Generate_{Subsets}$ */
11         Decomposed $G_S$ :
12         Basics;
13         Contents;
14         Connections;
15         Time;
16         Apply Transformer-Transfer-Learning
             $(G_S) = T_F$ ;  /* where $T_F$ stands for
             Train feature */
           ;
17         Apply SMOTE $(T_F) = B_F$ ;  /* where $B_F$
             stands for Balancing Features */
           ;
18         Apply $CNN(B)_F) = (DF)_E$ ;    /* where
             $DF_E$ stands for Deep Features
             Extraction */
           ;
       else
           Got to Step 10;    /* repeat again and
             again */
19 Get $(DF)_E$;
20 Apply $IDS - INT((DF)_E) = D2AT$ ;    /* DDoS,
     analysis, backdoor, fuzzers, etc.  */
21 Finish;
```

4. Results and discussions

4.1. Dataset preparation

The proposed approach is tested extensively using three standard datasets, i.e. UNSW-NB15 [35,36], CIC-IDS2017 [37], NSLKDD Dataset 3 [38]. To prepare the UNSW-NB15 dataset, the packet parser is used to collect 100 GB of raw traffic in the form of PCAP files. This dataset contains nine attack families, namely fuzzers, analysis, backdoors, DoS, exploits, generic, reconnaissance, shellcode, and worms. Table 2 shows the type of attack, the number of records, and information about each

Table 2
UNSW-NB15 dataset.

Type	No. of records	Information
Normal	2,218,761	Clean data exchange
Fuzzers	24,246	Randomly feeding a program or network to suspend it
Analysis	2,677	It has port scan, spam, and HTML file attacks
Backdoors	2,329	Stealth access to a computer or its data by bypassing system security
DoS	16,353	Malicious attempt to make a server or network resource unavailable to users
Exploits	44,525	The attacker exploits a software or operating system security flaw
Generic	215,481	A technique works against all block-ciphers with the given block and key size
Reconnaissance	13,987	Contains all information-gathering strikes
Shellcode	1,511	Payload is used to exploit a software vulnerability
Worms	174	The attacker exploits PC security flaws to replicate

attack. Worms is a minor class, i.e. 174, whereas generic attack types are a major class, i.e. 215, 481. In addition, the features are not categorized as equal number instances. This class imbalance is a significant issue that can affect intrusion detection accuracy. This dataset contains a variety of malicious attack features, making it difficult to analyze all of them at once. Therefore, it is further divided into four sub-datasets to test the proposed approach based on the nature of different types of features. These are basic, contents, connection, and time. Each subset includes features that are most relevant to the other. For instance, the time features *stime*, *ltime*, and *sintpkt* denote the start time, last time, and source inter-packet arrival time, respectively. Table 3 shows a chunk of features included in each subset of UNSW-NB15 dataset. The CICIDS2017 includes normal and ongoing intrusion attempts in the form of PCAP files, which represent real-world data. CICFlowMeter is a traffic flow generator and anlyzer that can be used for network activity assessment with categorized streams according to the time stamp, source as well as destination IPs, ports, protocols, etc. The network is analyzed weekly and some statistics are discovered, such as (Monday, normal activity), (Tuesday, attacks + normal activity), (Wednesday, attacks + normal activity), (Thursday, attacks + normal activity), and (Friday, attacks + normal activity). Furthermore, these attacks are of different types, such as brute force, DDoS, web attacks, infiltration, and port scan. Table 4 shows detailed information about CIC-IDS2017, including attack type, attacker, victim, and date and time. The NSL-KDD dataset addresses some of KDD CUP99's issues. NSL-KDD train and test sets have rational data. This allows testing the entire set without randomly selecting a small portion, as shown in Table 5. For instance, this dataset is further divided into four subsets namely, KDDTrain+, KDDTest+, KDDTest-21, KDDTrain+_20%. Therefore, research evaluations can be easily analyzed and compared with subsets of features. Because the KDD dataset has a lot of duplicate data, learning models tend to focus on frequent data. This prevents them from learning less common records, such as U2R and R2L attacks, which are more dangerous to network systems. The recurring data in the test set will also bias the assessment results, with better detection rates on frequent records. It includes DoS, probe, R2L, and U2R attacks.

4.2. Performance indicators

We used five different types of evaluation metrics to evaluate the proposed approach thoroughly: Precision, Recall, F1-score, Accuracy, and confusion matrix. True Positives (TP) and True Negatives (TN) are correctly classified as the proportion of normal and abnormal network traffic. Similarly, a large proportion of normal and abnormal network traffic is classified incorrectly as False Positives (FP) and False Negatives (FN). The effectiveness of general classification is assessed using Accuracy. Accuracy can be computed as the number of instances correctly classified divided by the total number of instances. The confusion matrix

Table 3
A chunk of four subsets, i.e., basic, contents, connections, and time categories extracted from UNSW-NB15 dataset.

Flow Category	Features	Information
Basic	dur	Record total duration
	sbytes	Source to destination bytes
	dbytes	Destination to source bytes
	sttl	Source to destination time to live
	dttl	Destination to source time to live
	sload	Source bits per second
	dload	Destination bits per second
Contents	swin	Source TCP window advertisement
	dwin	Destination TCP window advertisement
	stcpb	Source TCP sequence number
	dtcpb	Destination TCP sequence number
	smeansz	Mean of the flow packet size transmitted by the source
	dmeansz	Mean of the flow packet size transmitted by the destination
Connection	ct_srv_src	No. of connections that contain the same service and source address
	ct_srv_dst	No. of connections that contain the same service and destination address
	ct_dst_ltm	No. of connections of the same destination address
	ct_src_ltm	No. of connections of the same source address
	ct_src_dport_ltm	No: of connections between the same source address and the destination port
Time	sjit	Source jitter (m/sec)
	djit	Destination jitter (m/sec)
	stime	record start time
	ltime	record last time
	sintpkt	Source inter-packet arrival time (m/sec)

Table 4
CIC-IDS2017 dataset.

Attack	Attacker	Victim	Date
Brute Force	Kali, 205.174.165.73	Web Server Ubuntu, 205.174.165.68	July 4, 2017
DDoS	Kali, 205.174.165.73	Web Server Ubuntu, 205.174.165.68	July 5, 2017
Web Attack	Kali, 205.174.165.73	Victim: Web Server Ubuntu, 205.174.165.68	July 6, 2017, morning
Infiltration	Kali, 205.174.165.73	Windows Vista, 192.168.10.8	July 6, 2017, afternoon
Port Scan	Kali, 205.174.165.73	Ubuntu16, 205.174.165.68	Friday, July 7, 2017
Normal	–	–	Monday, July 3, 2017

Table 5
NSL-KDD dataset.

Data Files	Description
KDDTrain+	The full NSL-KDD train set
KDDTest+	The full NSL-KDD test set
KDDTest-21	A subset of the KDDTest+.arff file which does not include records with the difficulty level of 21 out of 21
KDDTrain+_20%	A 20% subset of the KDDTrain+.arff file

shows network traffic instances correctly or incorrectly classified as normal or abnormal. The evaluation matrices are shown in Equations (4)–(6)

$$Recall = \frac{FP}{(FP + TN)} \tag{4}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{5}$$

$$F1 - score = \frac{(2*TP)}{(2TP + FP + FN)} \tag{6}$$

$$Accuracy = \frac{(TP + TN)}{(TP + TN + FP + FN)} \tag{7}$$

4.3. Results analysis and comparison

To test and compare the proposed research, we developed three commonly used hybrid deep learning models: CNN-LSTM, CNN-Recurrent Neural Network (CNN-RNN), and CNN-Gated Recurrent Unit (CNN-GRU). Fig. 6 shows the dynamic accuracy epoch curves for the UNSW-NB15 dataset during training and testing, respectively. Because this dataset is divided into four subsets based on the nature of the network flows, the proposed IDS-INT is tested on all four subsets. For instance, the accuracy curves for four subsets using CNN-LSTM are shown in a)-d). CNN-RNN appears in e)-h) for the same four subsets, whereas CNN-GRU appears in i)-l). The red color represents the train, while the blue color represents the test data. It can be seen that curves with a basic subset using CNN-LSTM outperform others. For instance, the training curve begins at 10% while the test curve begins at 55% and gradually increases to 93% on the 20th epoch. Following that, both curves began to bend with each epoch and became more or less constant when 99% is been reached on the 35th epoch. The curves are quite close to each other and behave normally, indicating that there is no overfitting problem. In comparison to CNN-GRU, CNN-RNN is the next model that provides good detection results. CNN-RNN accuracy curves are quite dynamic, but the training and testing curves behave similarly. For instance, in part e), the training and testing curves begin at 90%, increase to 99%, and then suddenly drop to 97% on the 4th epoch. With abruptions, both curves behave similarly. Because of this discontinuity, the overall detection rates are lower than those of CNN-LSTM. The accuracy curves for CNN-RNN and CNN-GRU fall between 8% and 98%, respectively. Overall, CNN-LSTM provides the best epoch curves for accuracy values using four subsets of data.

Fig. 7 illustrates the loss epoch curves for training and test data derived from the four subsets of the UNSW-NB15 dataset.

The loss curves for four subsets using CNN-LSTM are shown in a)-d). For the same four subsets, CNN-RNN appears in e)-h) and CNN-GRU in i)-l). Training data is represented by yellow, while the test data is represented by green. The epoch loss curves using CNN-LSTM using the basic subset appear to be better because both the training and testing curves behave quite similarly as compared to the other three subsets. For instance, at first, both curves are greater than 100%, but by the 15th epoch, they have dropped to 40%. Following that, they dropped by 18% on the 20th epoch. On the 35th epoch, the test curve is increased to 40% and the training curve is reduced to 8%. CNN-RNN model is the next best, providing steady curves with lower loss values. CNN-RNN offers up to 5% epoch while CNN-GRU offers a 10% loss. Overall, CNN-LSTM provides the best epoch curves for loss values using four subsets of data.

Table 6 compares the performance of the four subsets of the UNSW-NB15 dataset. Three different hybrid deep learning models are used to investigate performance measures. It can be seen that using the basic subset with CNN-LSTM produces better results than the other three subsets. For instance, Precision, Recall, F1-score, and detection accuracy are all 99%, 100%, 99%, and 99.21%, respectively. CNN-LSTM with connection subset achieves Precision, Recall, F1-score, and Accuracy performance measures of 98%, 99%, 99%, and 98.92%, respectively. The CNN-RNN deep learning model comes next, with good detection results, while CNN-GRU has the worst. When compared to CNN-RNN and CNN-GRU, CNN-LSTM provides the highest values for the given performance measures. Fig. 8 depicts performance measures for various types of

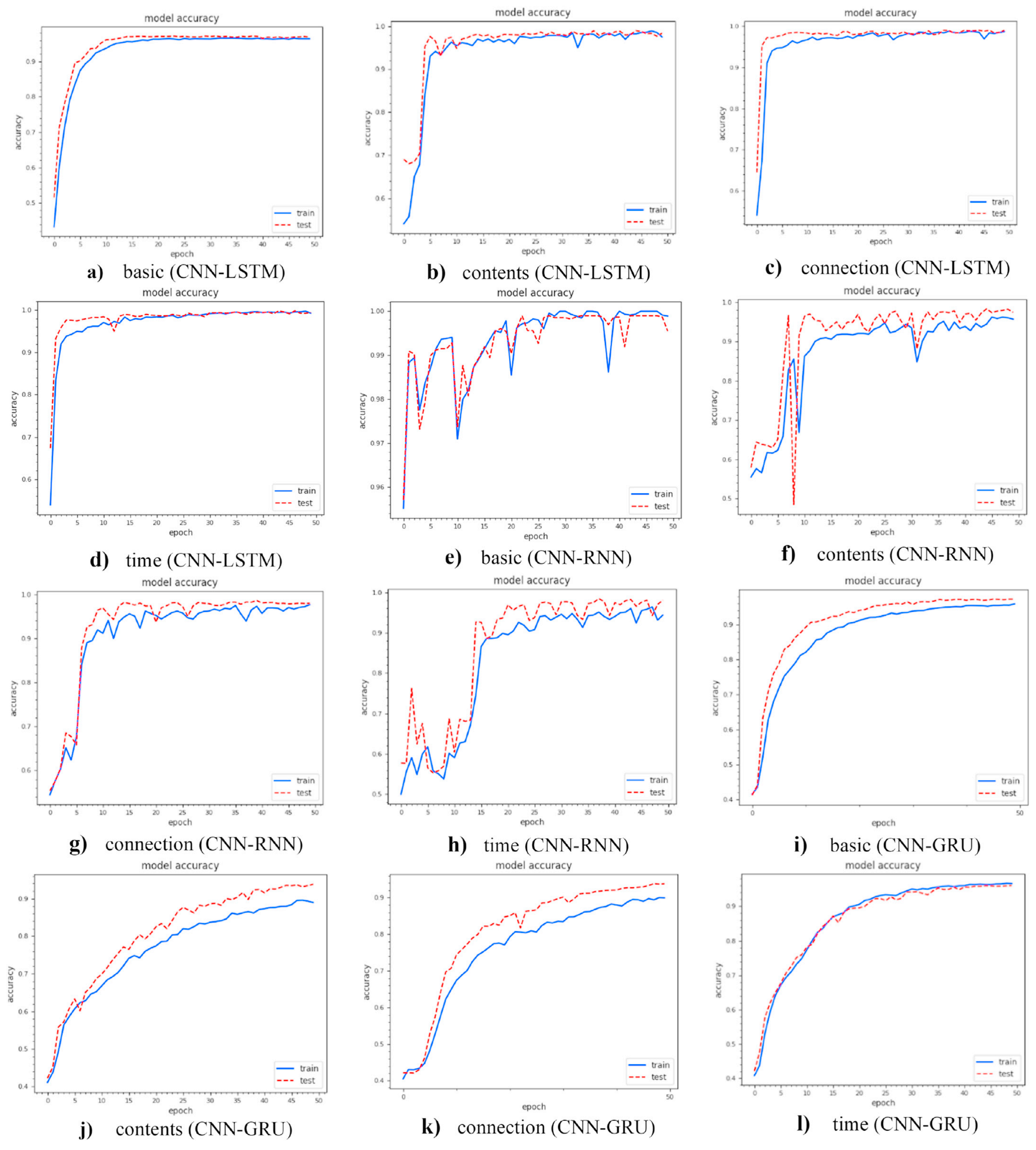


Fig. 6. Comparison of accuracy epoch curves for train and test data using the UNSW-NB15 dataset.

attacks using the UNSW-NB15 dataset. Blue, orange, and grey colors are used to represent Precision, Recall, and F1-score, respectively. When compared to Recall and F1-score, Precision is generally higher for normal and most attacks. Normal has 97%, analysis has 98%, exploits have 100%, generic has 99%, and so on. Fuzzers and exploits have the lowest Recall, while DoS and generic have the highest. This experiment shows how effective the proposed approach is for each type of attack.

The performance measure for each type of attack is shown in Table 7 using the CIC-IDS2017 dataset. Web attacks have the highest Precision, Recall, F1-score, and Accuracy of 99%, 100%, 99%, and 99.32%, respectively when compared to other attacks. The portscan attack is the next most correctly detected, with Precision, Recall, F1-score, and Accuracy scores of 99%, 100%, 99%, and 99.23%, respectively. The next correctly detected attack is an infiltration attack, followed by a DDoS attack. As can be seen, CNN-RNN is the next best performer in terms of four different types of attacks, while CNN-GRU is the worst. Table 8

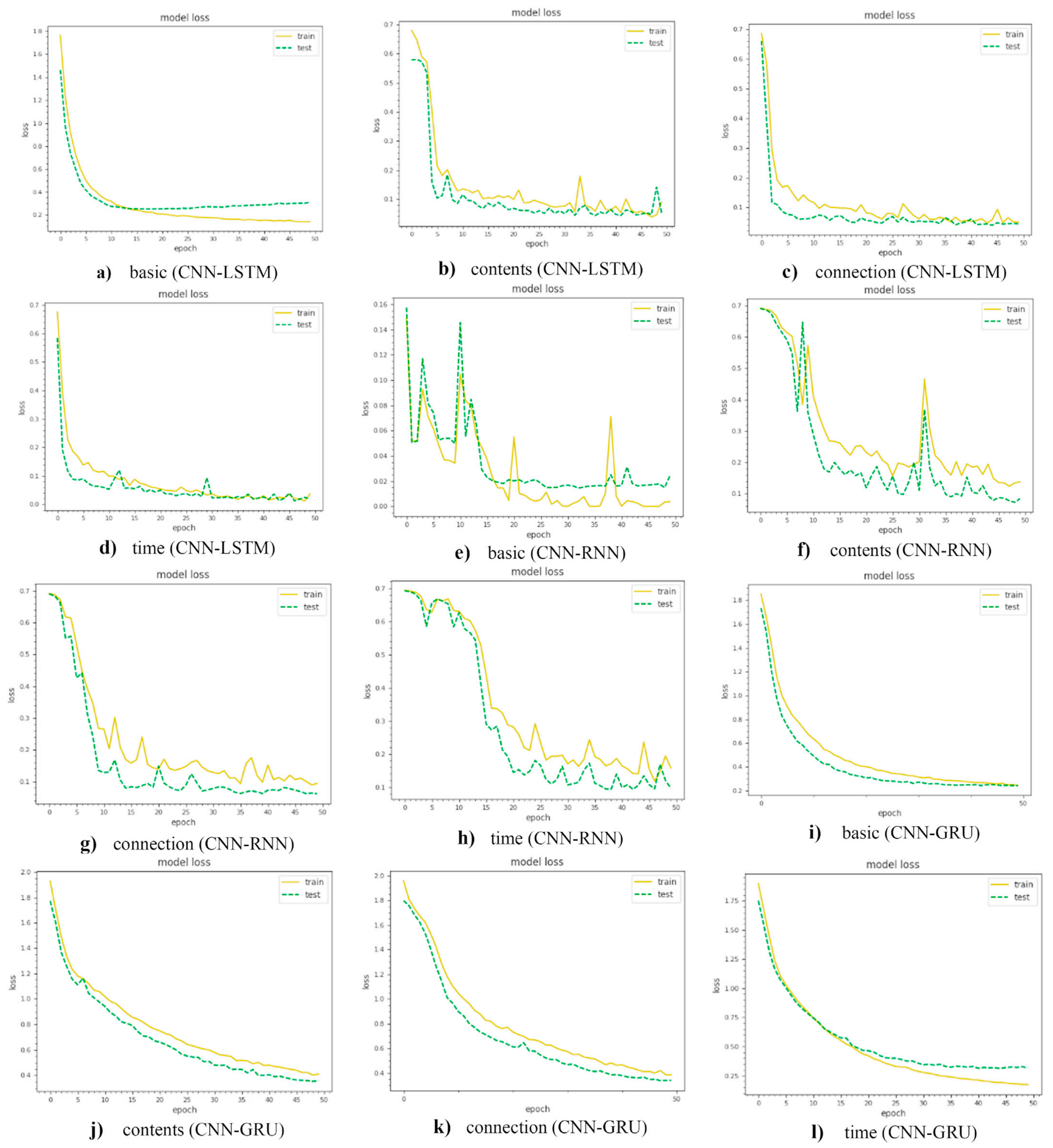


Fig. 7. Comparison of loss epoch curves for train and test data using the UNSW-NB15 dataset.

shows the performance measures for four different subsets extracted from the NSL-KDD dataset. The proposed IDS-INT achieves the best detection results when using CNN-LSTM. For instance, with the KDDTest + subset, Precision is 98%, Recall is 99%, F1-score is 98%, and Accuracy is 98.45%. When compared to CNN-RNN and CNN-GRU, these are the best detection results.

Fig. 9 depicts the classification and misclassification matrices for each type of attack using the UNSW-NB15 dataset. The dataset is divided into four subsets, each of which contains nine different types of attack classes and one normal class. The diagonal values represent the correct classification values, whereas the off-diagonal values represent the misclassification results. For each type of attack, CNN-LSTM with a basic subset has the highest classification values. For instance, 97%, 98%, 98%, 97%, 94%, 97%, 100%, 99%, and 98% for analysis, backdoor, DoS, exploits, fuzzers, generic, reconnaissance, shellcode, and worms, respectively. As can be seen, CNN-GRU with connection subset has the lowest detection

Table 6
Comparison of performance measures using the UNSW-NB15 dataset.

Models	Features	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
CNN-LSTM	basic	99	100	99	99.21
	connection	98	99	99	98.92
	contents	99	98	99	98.60
	time	98	99	100	98.52
CNN-RNN	basic	97	97	96	97.14
	connection	96	97	98	96.76
	contents	96	96	98	96.88
	time	97	98	98	97.43
CNN-GRU	basic	95	94	95	94.20
	connection	94	95	94	94.18
	contents	93	93	94	93.71
	time	93	93	92	92.62

Table 7
Comparison of performance measures using the CIC-IDS2017 dataset.

	Attacks	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
CNN-LSTM	DDoS	99	99	99	98.96
	PortScan	99	100	99	99.23
	Infiltration	100	98	99	99.12
	WebAttacks	99	100	99	99.32
CNN-RNN	DDoS	98	98	96	97.45
	PortScan	97	96	97	96.88
	Infiltration	97	97	96	96.72
	WebAttacks	98	98	97	97.94
CNN-GRU	DDoS	96	97	95	96.4
	PortScan	96	96	96	95.76
	Infiltration	95	95	94	95.1
	WebAttacks	98	97	97	97.24

results. For instance, 77%, 87%, 83%, 100%, 64%, 71%, 99%, 96%, 80%, and 96% for analysis, backdoor, DoS, exploits, fuzzers, generic, reconnaissance, shellcode, and worms, respectively. Overall, CNN-LSTM has the best classification for all four subsets, with CNN-RNN coming in second and CNN-GRU coming in last.

The confusion matrices for four subsets of the NSL-KDD dataset are shown in Fig. 10. Each subset contains four distinct attacks: DoS, probe, U2R, and R2L, and one benign. For benign, DoS, probe, U2R, and R2L, the proposed approach with the KDDTrain + subset has classification values of 98%, 99%, 98%, 97%, and 99%, respectively. Similarly, the classification values for benign, DoS, probe, U2R, and R2L in KDDTest + are 97%, 99%, 99%, 97%, and 98%, respectively. Using four different subsets of the NSL-KDD dataset, it is demonstrated that the proposed approach provides the highest classification values for each type of attack. In [13], the authors used Android network traffic features to classify malware. The authors used the BERT base model with the traditional machine learning methods to detect Android malware with a classification accuracy of 99%. The proposed work used the advanced version of BERT, i.e., BERT large, with the combined approach of deep learning methods such as CNN-LSTM to develop IDS-INT. This method detects different types of intrusions with an accuracy of 99.21%. It demonstrates that increasing the size of the model from BERT base to BERT large using deep learning techniques improves performance.

Table 8
Comparison of performance measures using the NSL-KDD dataset.

	Data	Precision (%)	Recall (%)	F1-score (%)	Accuracy (%)
CNN-LSTM	KDDTrain+	99	98	97	98.1
	KDDTest+	98	99	98	98.45
	KDDTest-21	97	99	98	98.32
	KDDTrain+_20%	98	96	98	97.81
CNN-RNN	KDDTrain+	97	96	96	96.22
	KDDTest+	96	97	97	96.1
	KDDTest-21	95	96	95	95.38
	KDDTrain+_20%	97	95	95	95.28
CNN-GRU	KDDTrain+	96	95	95	95.1
	KDDTest+	95	94	95	94.62
	KDDTest-21	95	93	94	94.22
	KDDTrain+_20%	94	94	93	93.96

4.4. Model interpretation using explainable AI

Explainable AI technology evaluates model input features and identifies the features that drive the model. It gives us a sense of control because we can choose whether or not to trust the predictions of these models. An explainable AI strategy is employed to interpret and evaluate the proposed IDS-INT. The Local Interpretable Model-agnostic Explanation

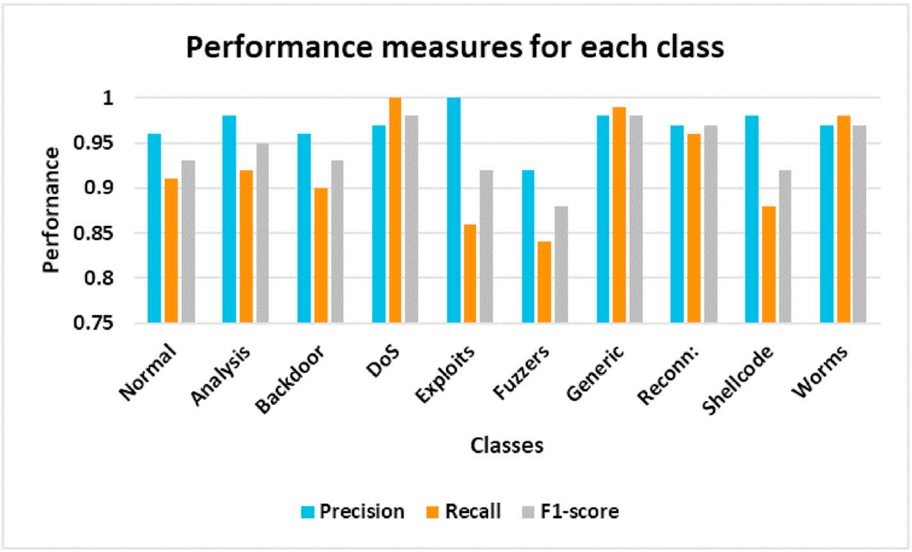


Fig. 8. Performance measures for different types of attacks using the UNSW-NB15 dataset.

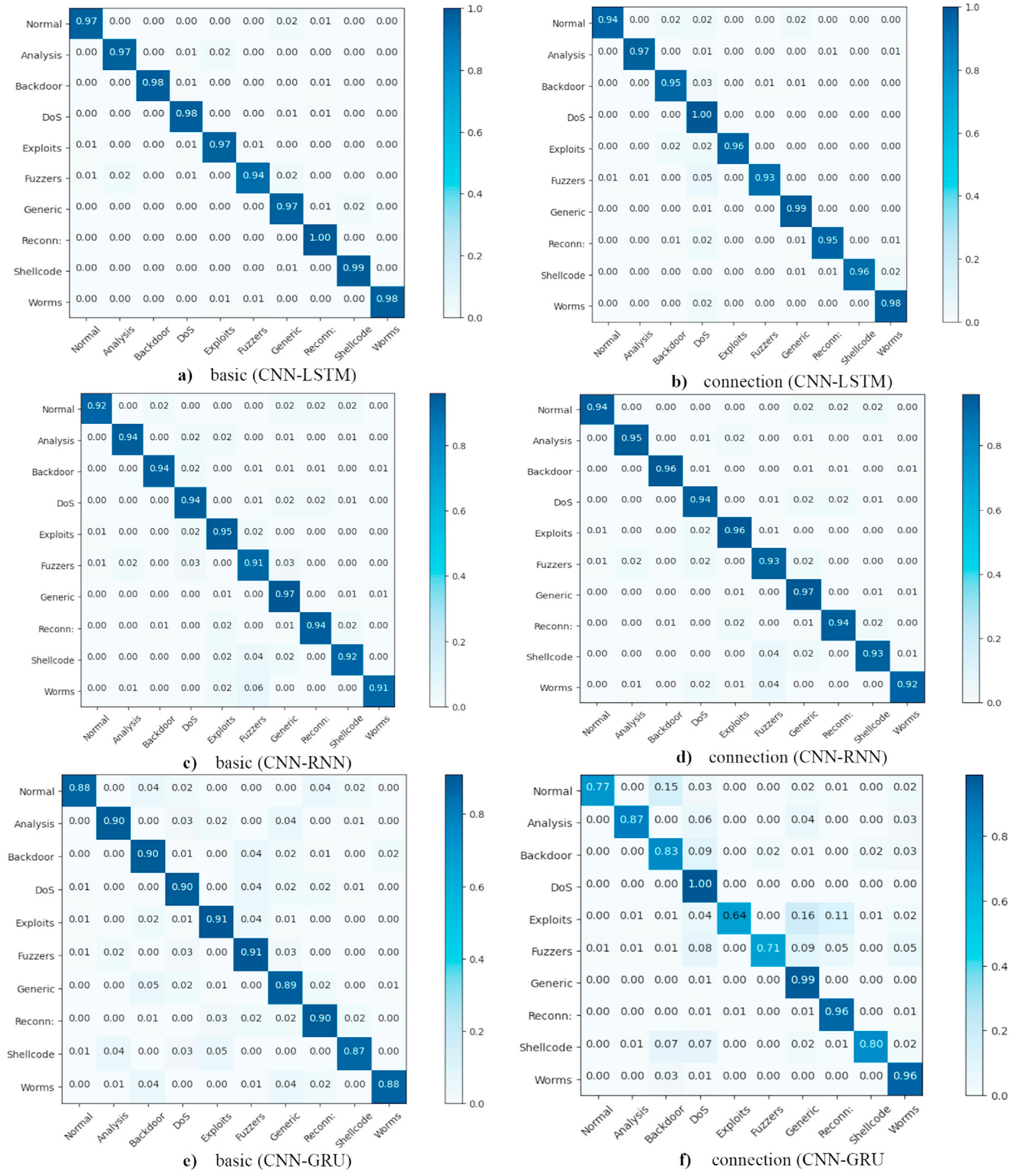


Fig. 9. Comparison of confusion matrices using the UNSW-NB15 dataset.

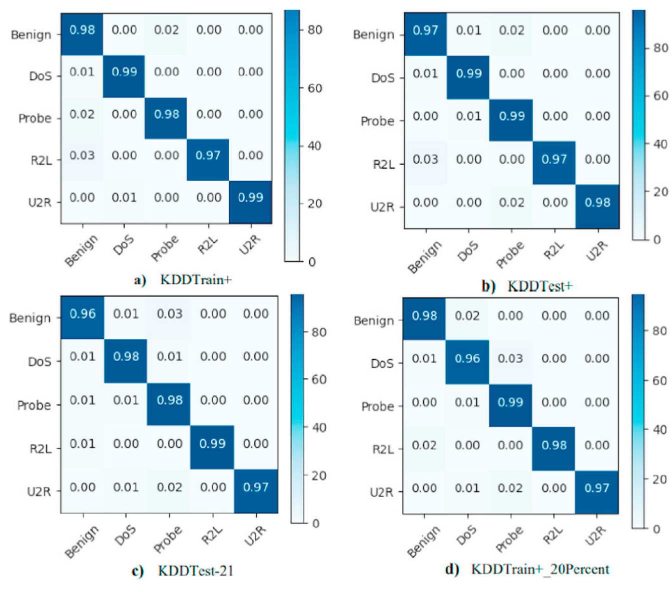


Fig. 10. Comparison of confusion matrices using the NSL-KDD dataset.

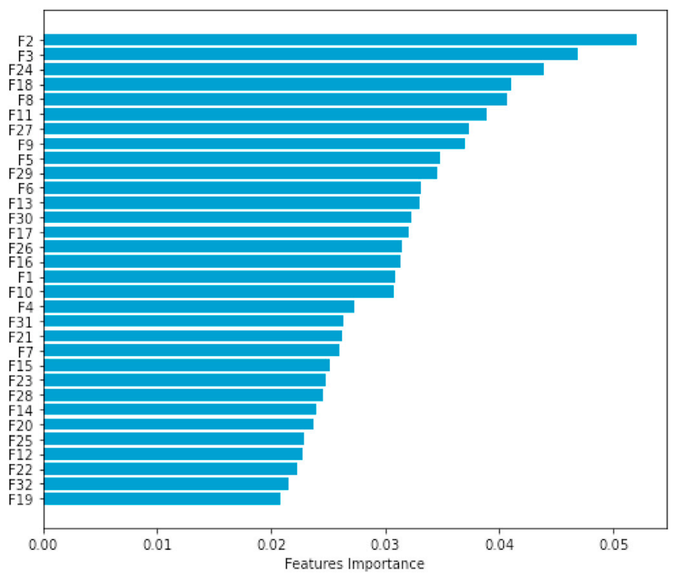


Fig. 11. A chunk of feature with their importance level.

(LIME) and Shapley Additive exPlanations (SHAP) libraries are utilized to explain the effect of each feature on the accuracy of the IDS-INT [39]. There are a total of 49 features in the UNSW-NB15 dataset, and we need to demonstrate the impact of features among each other to show the effectiveness of our approach. We selected a subset of 32 features to better visualize and analyze the impact of features on output behavior. This experiment exhibits the most valuable features out of a set of 32 features. The features are presented in the following order: (F1: source IP, F2: source port, F3: destination IP, F4: destination port, F5: transaction protocol, F6: state of the protocol, F7: duration, F8: source-to-destination transaction bytes, F9: destination-to-source transaction bytes, F10: source-to-destination TTL, F11: destination-to-source TTL, source packets retransmitted, F12: destination packets retransmitted, F13: protocol service, F14: source bits per second, F15: destination bits per second, F16: source-to-destination packet count, F17: destination-to-source packet count, F18: source TCP window, F19: source window advertisement, F20: destination window advertisement, F21: source sequence number, F22: destination sequence number, F23: mean packet size from source, F24: mean packet size from destination, F25: pipelined depth into the connection, F26: uncompressed content size, F27: source jitter, F28: destination jitter, F29: record start time, F30: record last time, F31: source inter-packet arrival time, F32: destination inter-packet arrival time). The relative strength of the first 32 features is depicted in Fig. 11. For instance, the F2 feature is the most important, while the F19 feature is the least important of the 32 features. The F3 and F24 features are the next most important. By doing so, we can extract the most important features of the proposed approach one by one. The SHAP values represent the level of support that a feature provides for a model result. Fig. 12 depicts how feature SHAP values influence output data ranging from prior expectations to detection accuracy. SHAP attributes are used to order the features, with the lowest values at the bottom of the peak showcase. The colors red and blue represent the contributions of abnormal (attacks) and normal (benign) features, respectively. There are 32 features in total, with the top nine most powerful ones highlighted. The F1 feature has a value of 244 and is colored red, indicating that it is the most important contributor to detecting abnormal network behavior. Overall, the features F1, F19, F8, F28, F5, and F3 can contribute to abnormal traffic. Whereas, F16, F17, and F20, can contribute to normal traffic.

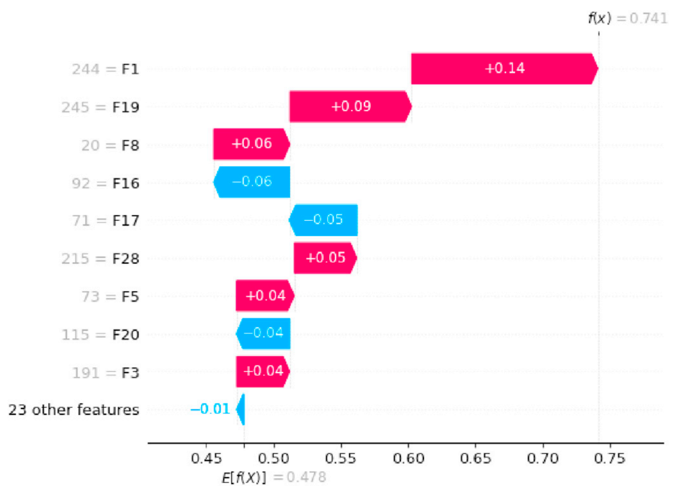


Fig. 12. The waterfall plot compares feature SHAP values to data distribution.

Fig. 13 depicts the importance of features in obtaining abnormal traffic from a set of measurements with a threshold level of normal traffic. The 0.74 threshold divides feature influence into two categories: normal and abnormal network traffic. The colors red and blue represent the presence of features in abnormal and normal traffic, respectively. It is discovered that the F1 feature contributed significantly to abnormal network traffic while the F17 feature contributed significantly to normal network traffic. This allows us to quickly identify the features that contribute significantly to each type of network traffic. Fig. 14 illustrates the influence of selected features on predicted value. The red color reflects the significant response of each feature, whereas the green color reflects the lesser impact. As can be seen, the cumulative impact of the F2 feature is substantial, whereas F20 is the least. This makes it simple to characterize the influence of each feature for a specific network type, such as normal or abnormal. This test demonstrates how each feature affects a specific type of network traffic. Further, to prevent positive and negative SHAP values from canceling each other, we use absolute values. Fig. 15 captures the main and interaction effects of the initial ten

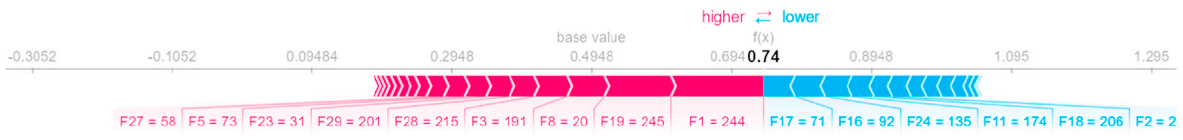


Fig. 13. Feature contribution to a class based on a threshold using SHAP force plot.

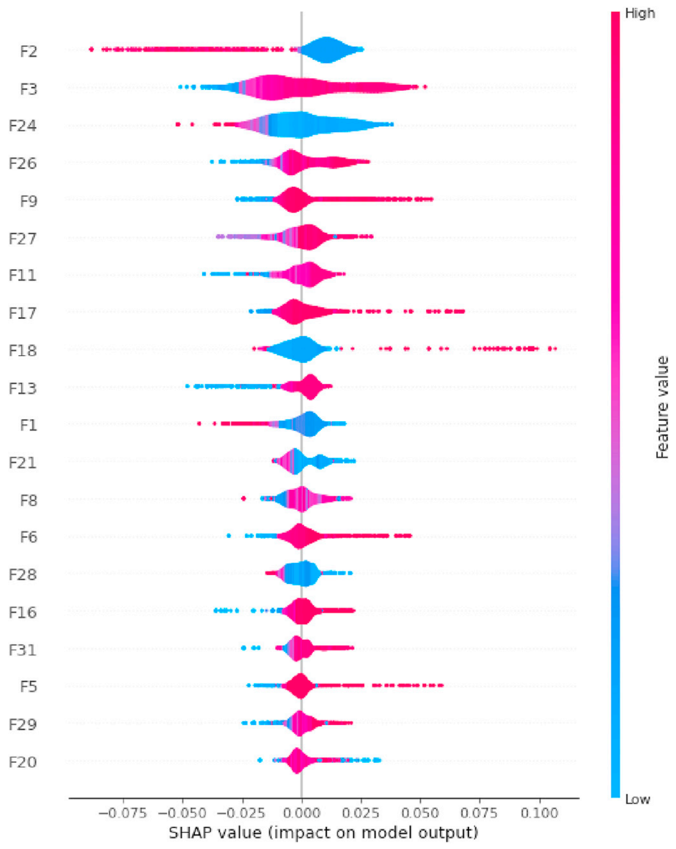


Fig. 14. The combined feature importance with feature effects.

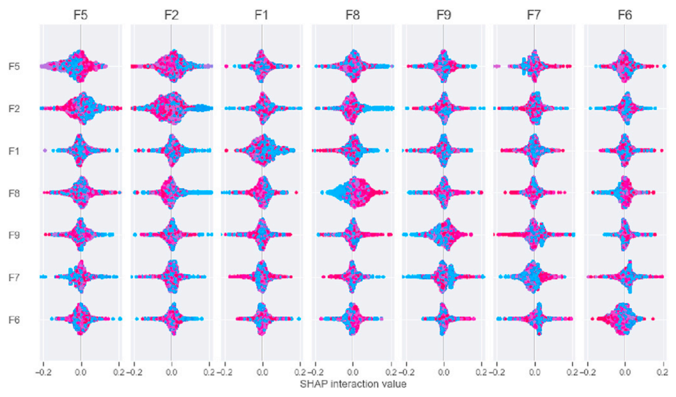
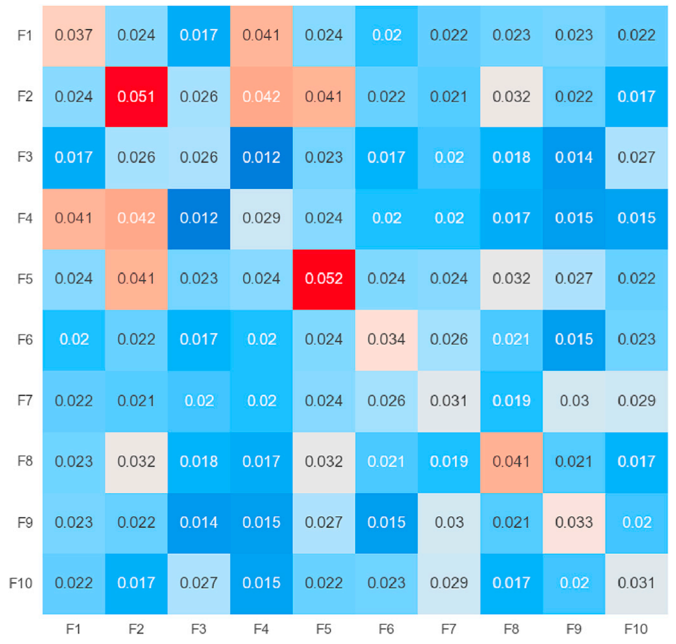


Fig. 16. Feature interaction for model output.



	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
F1	0.037	0.024	0.017	0.041	0.024	0.02	0.022	0.023	0.023	0.022
F2	0.024	0.051	0.026	0.042	0.041	0.022	0.021	0.032	0.022	0.017
F3	0.017	0.026	0.026	0.012	0.023	0.017	0.02	0.018	0.014	0.027
F4	0.041	0.042	0.012	0.029	0.024	0.02	0.02	0.017	0.015	0.015
F5	0.024	0.041	0.023	0.024	0.052	0.024	0.024	0.032	0.027	0.022
F6	0.02	0.022	0.017	0.02	0.024	0.034	0.026	0.021	0.015	0.023
F7	0.022	0.021	0.02	0.02	0.024	0.026	0.031	0.019	0.03	0.029
F8	0.023	0.032	0.018	0.017	0.032	0.021	0.019	0.041	0.021	0.017
F9	0.023	0.022	0.014	0.015	0.027	0.015	0.03	0.021	0.033	0.02
F10	0.022	0.017	0.027	0.015	0.022	0.023	0.029	0.017	0.02	0.031

Fig. 15. Absolute means of main effects and their interactions for the first 10 features.

features. For instance, it can be seen that the primary average impact for F1, F2, F5, and F8 is greater. These features are more likely to have major negative or positive impacts. In other words, these traits have a significant impact on the detection of different types of network traffic. The interaction effects of the F6, F7, and F9 are also considerable. Fig. 16 captures the contact of features relating to final detection. On the diagonals, we see the SHAP values for the main effects, whereas the off-diagonals display the interaction effects. The main and interaction effects are highlighted, which can shed light on absolute mean values. High absolute mean values are associated with high SHAP values. Analyzing the connections shown by the interaction values provides further insight. For instance, F1, F2, F5, and F8 features have greater effects.

5. Conclusion

An intelligent network intrusion detection system can protect against different types of malicious attacks. The network behavior representation includes a wide range of features such as attack reference, attack and host details, malicious script, etc. When compared to typical network trends, network traffic can have an imbalanced variety of hazardous intrusions. It is challenging to develop a transfer learning-based method for detecting a particular type of network attack due to complex attributes and data imbalance issues. To address these concerns, this study proposes the IDS-INT method for detecting various types of attacks using imbalanced network traffic. The transfer learning method is used by IDS-INT to discover feature relationships in data imbalance and network representation learning. The network interaction descriptions—including network nodes, attack type, reference, host information, etc.—are used to gather attack details. To discover the specific feature map with the help of their contextual anchors. Next, the SMOTE method is implemented to balance the network traffic by learning more about minority attacks. The CNN model has been developed for dee-featuring extraction from balanced network data. The CNN-LSTM model is designed to learn the deep features for detecting different types of attacks. We have analyzed the proposed IDS-INT with three big and standard datasets namely CIC-IDS2017, UNSW-NB15, and NSL-KDD. Further, baseline experiments are conducted with CNN-RNN and CNN-GRU to compare the proposed IDS-INT. Our approach outperformed the baseline methods with 99% Precision, 100% Recall, 99% F1-score, and 99.21% Accuracy. To interpret the proposed IDS-INT and develop a trustworthy model, an explainable AI approach has been implemented. The real-time network can be extremely dynamic and imbalanced. Furthermore, it may contain malicious scripts and attack references that move from random clients to the server. The proposed method helped to address the problem of imbalanced data traffic and overfitting. The transformer-based transfer learning approach is used to extract meaningful information from transmitted data between network nodes. Furthermore, the proposed method effectively interprets and explains the impact and features to develop a trustworthy and reliable model.

Our proposed IDS-INT method can be used on a variety of mobile devices, including Android and Windows-based network edges. In the future, we will try to develop a client-server communication model using an advanced deep learning approach, i.e., federated learning, to detect various types of attacks on network edges.

Data availability

Data that support the findings of this study are openly available: UNSW-NB15,[1] CIC-IDS2017,[2] NSL-KDD[3]

References

[1] B.B. Zarpelão, R.S. Miani, C.T. Kawakani, S.C. de Alvarenga, A survey of intrusion detection in internet of things, J. Netw. Comput. Appl. 84 (2017) 25–37.

[2] R. Samrin, D. Vasumathi, Review on anomaly based network intrusion detection system, in: 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), IEEE, 2017, pp. 141–147.

[3] S.R. Davies, R. Macfarlane, W.J. Buchanan, Differential area analysis for ransomware attack detection within mixed file datasets, Comput. Secur. 108 (2021) 102377.

[4] J. Liu, Y. Gao, F. Hu, A fast network intrusion detection system using adaptive synthetic oversampling and lightgbm, Comput. Secur. 106 (2021) 102289.

[5] T. Mehmood, H.B.M. Rais, Machine learning algorithms in context of intrusion detection, in: 2016 3rd International Conference on Computer and Information Sciences (ICCOINS), IEEE, 2016, pp. 369–373.

[6] N. Shone, T.N. Ngoc, V.D. Phai, Q. Shi, A deep learning approach to network intrusion detection, IEEE transactions on emerging topics in computational intelligence 2 (1) (2018) 41–50.

[7] B.B. Rao, K. Swathi, Fast knn classifiers for network intrusion detection system, Indian Journal of Science and Technology 10 (14) (2017) 1–10.

[8] L. Koc, T.A. Mazzuchi, S. Sarkani, A network intrusion detection system based on a hidden naïve bayes multiclass classifier, Expert Syst. Appl. 39 (18) (2012) 13492–13500.

[9] S. Sahu, B.M. Mehtre, Network intrusion detection system using j48 decision tree, in: 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2015, pp. 2023–2026.

[10] Y. Chang, W. Li, Z. Yang, Network intrusion detection based on random forest and support vector machine, in: 2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), vol. 1, IEEE, 2017, pp. 635–638.

[11] A. Rosay, F. Carlier, P. Leroux, Mlp4nids: an efficient mlp-based network intrusion detection for cicids2017 dataset, in: International Conference on Machine Learning for Networking, Springer, 2019, pp. 240–254.

[12] C. Yue, L. Wang, D. Wang, R. Duo, X. Nie, An ensemble intrusion detection method for train ethernet consist network based on cnn and rnn, IEEE Access 9 (2021) 59527–59539.

[13] F. Ullah, A. Alsirhani, M.M. Alshahrani, A. Alomari, H. Naeem, S.A. Shah, Explainable malware detection system using transformers-based transfer learning and multi-model visual representation, Sensors 22 (18) (2022) 6766.

[14] Y.E. Seyyar, A.G. Yavuz, H.M. Ünver, Detection of web attacks using the bert model, in: 2022 30th Signal Processing and Communications Applications Conference (SIU), IEEE, 2022, pp. 1–4.

[15] K. Yu, L. Tan, S. Mumtaz, S. Al-Rubaye, A. Al-Dulaimi, A.K. Bashir, F.A. Khan, Securing critical infrastructures: deep-learning-based threat detection in iiot, IEEE Commun. Mag. 59 (10) (2021) 76–82.

[16] P. He, J. Zhu, S. He, J. Li, M.R. Lyu, Towards automated log parsing for large-scale log data analysis, IEEE Trans. Dependable Secure Comput. 15 (6) (2017) 931–944.

[17] J. Li, H. Zhang, Z. Wei, The weighted word2vec paragraph vectors for anomaly detection over http traffic, IEEE Access 8 (2020) 141787–141798.

[18] S. Huang, Y. Liu, C. Fung, R. He, Y. Zhao, H. Yang, Z. Luan, Hitanomaly: hierarchical transformers for anomaly detection in system log, IEEE transactions on network and service management 17 (4) (2020) 2064–2076.

[19] E. Min, J. Long, Q. Liu, J. Cui, W. Chen, Tr-ids: Anomaly-Based Intrusion Detection through Text-Convolutional Neural Network and Random Forest, Security and Communication Networks, 2018.

[20] B. Aslahi-Shahri, R. Rahmani, M. Chizari, A. Maralani, M. Eslami, M.J. Golkar, A. Ebrahimi, A hybrid method consisting of ga and svm for intrusion detection system, Neural Comput. Appl. 27 (6) (2016) 1669–1676.

[21] H. Alazzam, A. Sharieh, K.E. Sabri, A feature selection algorithm for intrusion detection system based on pigeon inspired optimizer, Expert Syst. Appl. 148 (2020) 113249.

[22] C. Khammassi, S. Krichen, A ga-lr wrapper approach for feature selection in network intrusion detection, Comput. Secur. 70 (2017) 255–277.

[23] S. Pouyanfar, S. Sadiq, Y. Yan, H. Tian, Y. Tao, M.P. Reyes, M.-L. Shyu, S.-C. Chen, S.S. Iyengar, A survey on deep learning: algorithms, techniques, and applications, ACM Comput. Surv. 51 (5) (2018) 1–36.

[24] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, S. Venkatraman, Deep learning approach for intelligent intrusion detection system, IEEE Access 7 (2019) 41525–41550.

[25] S. Jian, G. Pang, L. Cao, K. Lu, H. Gao, Cure: flexible categorical data representation by hierarchical coupling learning, IEEE Trans. Knowl. Data Eng. 31 (5) (2018) 853–866.

[26] S. Naseer, Y. Saleem, S. Khalid, M.K. Bashir, J. Han, M.M. Iqbal, K. Han, Enhanced network anomaly detection based on deep neural networks, IEEE Access 6 (2018) 48231–48246.

[27] F.A. Acheampong, H. Nunoo-Mensah, W. Chen, Transformer models for text-based emotion detection: a review of bert-based approaches, Artif. Intell. Rev. 54 (8) (2021) 5789–5829.

[28] A. Yates, R. Nogueira, J. Lin, Pretrained transformers for text ranking: bert and beyond, in: Proceedings of the 14th ACM International Conference on Web Search and Data Mining, 2021, pp. 1154–1156.

[29] A. Fernández, S. Garcia, F. Herrera, N.V. Chawla, Smote for learning from imbalanced data: progress and challenges, marking the 15-year anniversary, J. Artif. Intell. Res. 61 (2018) 863–905.

[30] M. Azizjon, A. Jumabek, W. Kim, 1d cnn based network intrusion detection with normalization on imbalanced data, in: 2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC), IEEE, 2020, pp. 218–224.

[31] Y. Li, Y. Xu, Z. Liu, H. Hou, Y. Zheng, Y. Xin, Y. Zhao, L. Cui, Robust detection for network intrusion of industrial iot based on multi-cnn fusion, Measurement 154 (2020) 107450.

[32] X. Zhang, J. Ran, J. Mi, An intrusion detection system based on convolutional neural network for imbalanced network traffic, in: 2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT), IEEE, 2019, pp. 456–460.

[33] R. Vinayakumar, K. Soman, P. Poornachandran, Applying convolutional neural network for network intrusion detection, in: 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), IEEE, 2017, pp. 1222–1228.

[34] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, J. Chen, Dl-ids: Extracting Features Using Cnn-Lstm Hybrid Network for Intrusion Detection System, Security and Communication Networks, 2020.

[35] N. Moustafa, J. Slay, Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set), in: 2015 Military Communications and Information Systems Conference (MilCIS), IEEE, 2015, pp. 1–6.

[36] N. Moustafa, J. Slay, The evaluation of network anomaly detection systems: statistical analysis of the unsw-nb15 data set and the comparison with the kdd99 data set, Inf. Secur. J. A Glob. Perspect. 25 (1–3) (2016) 18–31.

[37] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, ICISSp 1 (2018) 108–116.

[38] M. Tavallaee, E. Bagheri, W. Lu, A.A. Ghorbani, A detailed analysis of the kdd cup 99 data set, in: 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ieee, 2009, pp. 1–6.

[39] D.L. Marino, C.S. Wickramasinghe, M. Manic, An adversarial approach for explainable ai in intrusion detection systems, in: IECON 2018-44th Annual Conference of the, IEEE Industrial Electronics Society, IEEE, 2018, pp. 3237–3243.

[1] https://research.unsw.edu.au/projects/unsw-nb15-dataset.

[2] https://www.unb.ca/cic/datasets/ids-2017.html.

[3] https://www.unb.ca/cic/datasets/nsl.html).