Expert Systems With Applications 238 (2024) 121751

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach

Bhawana Sharma [a], Lokesh Sharma [a,*], Chhagan Lal [b], Satyabrata Roy [a]

[a] Manipal University Jaipur, Jaipur, Rajasthan, India
[b] Department of Intelligent Systems, Cybersecurity Group, TU Delft, Netherlands

ARTICLE INFO

Keywords:
Intrusion detection system
DL
Deep neural network
Convolution neural network
XAI
Local interpretable model-agnostic explanations
Shapley additive explanations

ABSTRACT

The Internet of Things (IoT) is currently seeing tremendous growth due to new technologies and big data. Research in the field of IoT security is an emerging topic. IoT networks are becoming more vulnerable to new assaults as a result of the growth in devices and the production of massive data. In order to recognize the attacks, an intrusion detection system is required. In this work, we suggested a Deep Learning (DL) model for intrusion detection to categorize various attacks in the dataset. We used a filter-based approach to pick out the most important aspects and limit the number of features, and we built two different deep-learning models for intrusion detection. For model training and testing, we used two publicly accessible datasets, NSL-KDD and UNSW-NB 15. First, we applied the dataset on the Deep neural network (DNN) model and then the same dataset on Convolution Neural Network (CNN) model. For both datasets, the DL model had a better accuracy rate. Because DL models are opaque and challenging to comprehend, we applied the idea of explainable Artificial Intelligence (AI) to provide a model explanation. To increase confidence in the DNN model, we applied the explainable AI (XAI) Local Interpretable Model-agnostic Explanations (LIME) method, and for better understanding, we also applied Shapley Additive Explanations (SHAP).

1. Introduction

In recent years, IoT has been gaining popularity, and with the advancement of technologies, the internet, and big data, security has become essential for IoT networks. Researchers are seeking attention to the intrusion detection system for IoT networks for detecting malicious activities. Identifying any suspicious or abnormal activity generates a signal, thus preventing vulnerable devices. Since many heterogeneous devices for different applications are connected in IoT networks and generate big data within the network, thus the significant challenges are storage, computation of big data, and cyber security in IoT networks (Al-Fuqaha, Guizani, Mohammadi, Aledhari, & Ayyash, 2015; Da Xu, He, & Li, 2014).

The Intrusion Detection System (IDS) has two types of detection methods. One method is Signature-based IDS which detects malicious activity based on known signatures stored in the database; another method is anomaly-based, which detects the abnormal behavior of the system.

Signature-based IDS are proven to be inefficient in today's scenario for two main reasons. First, it needs the predetermined knowledge of signatures or attacks and is thus incapable of detecting new or zero-day attacks. Secondly, storing attacks in the database and computation for the devices in IoT networks with limited storage and computation capacity is inefficient.

Anomaly-based IDS detects abnormal behavior and is thus capable of detecting new or unknown attacks which are different from normal ones. The drawback is that it detects any change from the normal behavior and identifies it as abnormal behavior, and thus false positives are generated (Ahmad, Shahid Khan, Wai Shiang, Abdullah, & Ahmad, 2021; Sharma, Sharma, & Lal, 2019). With recent development in Machine Learning (ML)/Deep Learning (DL) techniques, these techniques are employed in Anomaly-based IDS to remove the drawbacks. Anomaly-based detection using ML/DL techniques can detect intrusions with higher accuracy and is attracting many researchers for solutions in the direction of network security in IoT networks (Al-Garadi et al., 2020; Lin et al., 2017; Xin et al., 2018). Various attacks/threats occur daily in IoT systems, so there is a need to identify and mitigate the attacks to protect the network. In an IoT system, there are three layers first is perception; the network is middle, and the last is the application layer.

* Corresponding author.
E-mail addresses: bhawana.199308601@muj.manipal.edu (B. Sharma), lokesh.sharma@jaipur.manipal.edu (L. Sharma), c.lal@tudelft.nl (C. Lal), satyabrata.roy@jaipur.manipal.edu (S. Roy).

https://doi.org/10.1016/j.eswa.2023.121751
Received 15 June 2023; Received in revised form 22 August 2023; Accepted 19 September 2023
Available online 25 September 2023

- **Perception layer:** : In this layer, sensors/devices termed as things in IoT network collect the useful data, and then the data is transmitted to the network layer after processing. There are three major security issues disturbance of signals, tampering of hardware, and constrained IoT devices and sensors. Signals are transmitted via wireless technologies, so there is a risk of signal disturbance, and thus, the efficiency of signals is compromised. Furthermore, a physical attack can weaken hardware components because IoT devices and sensors function in an external environment. The third problem is that IoT devices and sensors' limited power consumption, storage, and processing power make them susceptible to attacks. Sleep deprivation, node capturing, data injection, eavesdropping, and interference by sending noise signals can affect confidentiality. We can mitigate these issues by encrypting data from one end to another.
- **Network Layer:** This layer transfers data using modern technologies, including Bluetooth, Zigbee, WIFI, and "Long-Term Evolution (LTE)" as well as cloud computing platforms, network gateways, routers, and switches. Eavesdropping, traffic analysis, and monitoring are three major security challenges in the network layer. Massive traffic overload has rendered the target inaccessible to authorized users. The device can shut down and stop working by DoS attacks and sinkhole attacks. Data secrecy can be hampered by Man-in-the-Middle (MitM) attacks. We can prevent eavesdropping and heavy traffic bombardment by using a network object with the appropriate protocols and software to monitor the network.
- **Application Layer:**This layer provides application-specific services to the system. It offers a range of applications where IoT systems are installed, such as smart parking, smart healthcare, smart homes, smart cities, etc. It monitors different applications and other layers of the IoT system.
  The main security issue in this layer is the authentication of different mechanisms used by various applications. IoT involves a lot of connected devices or things in the network, so there is a need to monitor shared data and manage the data. There are risks of phishing, malicious scripts, and SYN flooding. Different protocols, such as "Message Queuing Telemetry Transport (MQTT)" and "Constrained Application Protocol (CoAP)", are used to mitigate these issues (Al Nafea & Almaiah, 2021; Altulaihan, Almaiah, & Aljughaiman, 2022).

Nowadays, ML and DL techniques are widely used for anomaly-based detection, where models learn to determine the normal behavior in the training phase (Chaabouni, Mosbah, Zemmari, Sauvignac, & Faruki, 2019). Before applying ML and DL techniques, we intelligently select features to attain the maximum accuracy with the fewest features possible. By lowering the amount of features, we can speed up training, which lowers the cost of computation and storage.

In this paper, we employed DL models to detect intrusions based on anomaly detection, which depends on the behavior of the system. The DNN model classifies the normal/attack categories in multi-class classification. Since the ML and DL models are black boxes, we explain and interpret the models using the concept of explainable AI. Explainable AI is to explain and interpret the models and find what makes the model arrive at such predictions. Mostly ML and DL models are considered black boxes, and it is not easy to understand the models. Researchers are working in this direction to develop methods for explaining the black box ML and DL models. For greater comprehension and model explainability, the XAI approaches LIME and SHAP are frequently utilized today. The major contributions of this research document are:

1. Design a DL model to classify normal/attack categories in IoT networks.
2. The accuracy and computation speed of feature reduction utilizing a filter-based approach is improved with the fewest possible features.
3. Analyze the model, contrast it with other models, and fine-tune it using various hyper-parameters.
4. Explain the model using the concept of Explainable AI and identify important features and the effects of a feature on the prediction/detection results using LIME and SHAP.

The subsequent part of the paper is organized as follows: Section 2 describes the state-of-the-art review of ML/DL techniques used for detecting the intrusion and the concept of explainable AI. Section 3 details the research design and methodology, based on deep learning techniques, to classify normal/ attack classes in IoT networks. Section 4 shows the evaluation and analysis of the model's accuracy and evaluation metrics. Section 5 includes the model explanation. Section 6 insights into the future work and limitations of the model, and Section 7 summarizes the work done in the paper and includes the future work.

2. Related works

In this section, we conduct a study and present a systematic literature review providing the introspection about different ML/DL techniques based intrusion detection systems. With the vast expansion of IoT networks, security, and privacy are the prime areas which need to be considered. Researchers effectively analyze the network and identify different attacks to take measures to prevent the network. Deep learning and Machine Learning are widely used in the field of intrusion detection systems (Karatas, Demir, & Sahingoz, 2020; Khan & Herrmann, 2019; Ma, 2020).

2.1. IDS studies based on ML and DL techniques

With the recent advancement in ML/DL techniques, models constructed using these techniques are used by researchers for intrusion detection systems. Different ML models are applied by the researchers for intrusion detection, such as "K-Nearest Neighbor (KNN)" (Xu et al., 2018) and "Support Vector Machine (SVM)" (Teng, Wu, Zhu, Teng, & Zhang, 2017), and evaluated the models using KDD99, NSL-KDD, and DARPA datasets.

On openly accessible NSL-KDD[1] and UNSW-NB15[2] datasets, Fenanir, Semchedine, and Baadache (2019) implemented various ML-based models, utilized a filter approach to pick features, and used a Decision Tree (DT) to get the maximum accuracy. A lightweight IDS model was suggested by the author. The characteristics were selected using the filtering process, and the data was then categorized using ML techniques. The DT generates the most effective classification model on a variety of datasets, according to the experts' comparison of several machine learning techniques. The characteristics were selected using many datasets, various threshold values, and filter techniques such as the correlation filter methods like "Pearson Correlation Coefficient (PCC)", "Kendall Correlation Coefficient (KCC)" and "Spearman correlation coefficient (SCC)".[3] The author used a number of ML methods, including DT, SVM, and "Logistic Regression (LR)" for classification on multiple datasets, including UNSW-NB15, KDD99, and NSL-KDD.

Sun et al. (2020) created an LSTM-CNN model for classification using the hybrid method concept. To deal with the dataset's uneven distribution of the target class, the author employed the weight optimization strategy. The model's accuracy was tested using the CICIDS2017 dataset, and it was 98.67% accurate. Hassan, Gumaei, Alsanad, Alrubaian, and Fortino (2020) also suggested a hybrid deep neural network model that integrates CNN and LSTM. Model performance was evaluated based on accuracy parameters using the openly accessible UNSW-NB15 dataset, and an overall accuracy of 97% was attained.

[1] https://www.unb.ca/cic/datasets/nsl.html.
[2] https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecur.
[3] https://www.sciencedirect.com/topics/social-sciences/pearson-correlation-coefficient.

A CNN-based IDS model was proposed by Xiao, Xing, Zhang, and Zhao (2019), and the model performance was evaluated using the KDD Cup 99 dataset and found that it had a 94% accuracy rate. Denial of Service (DoS) attacks were the subject of Kim, Kim, Kim, Shim, and Choi (2020)'s CNN and RNN model, which had an accuracy of almost 99 percent on evaluating the model using KDD Cup dataset and similarly 91 percent on the CICIDS2018 dataset.

On the publicly accessible dataset UNSW-NB15, which was used to evaluate Kasongo and Sun (2020)'s DNN model, the model's accuracy for multi-class classification was 77.16%. On the publicly accessible dataset NSL-KDD cup, 97% accuracy was attained using the DNN model that Liang et al. (2019) presented. The DNN model was created by Thamilarasu and Chawla (2019), who then tested it on their own dataset and attained great precision. Ge, Syed, Fu, Baig, and Robles-Kelly (2021) developed the FNN model, evaluated it using the BoT-IoT dataset, and achieved a multi-class classification accuracy of above 99%. The DNN model was created by Vinayakumar et al. (2019), and it achieved a 78 percent accuracy rate on the publicly accessible dataset NSL-KDD.

Nagisetty and Gupta (2019) suggested many DL models, including CNN, DNN, MLP, and autoencoder. The models were tested using the open-source datasets UNSW-NB15 and NSL-KDD, and the DNN model outperformed them in terms of accuracy. Qiu et al. (2020) developed a DL-based model in which DoS assaults might be generated by a little modification in the characteristics. On the DMD-2018 dataset, the Vinayakumar et al. (2020)-proposed deep learning CNN and RNN models yielded 99% accuracy. A DNN model with varying numbers of neurons and hidden layers that are customized with different learning rates was proposed by the author. Model performance was evaluated on several publicly accessible datasets, including binary-class datasets with attack and normal classes and multi-class datasets with various attacks and normal classes. On the KDDCup99 dataset and the NSL-KDD dataset, the authors' model had an accuracy of 93% and 78%, respectively, after they reduced the number of features and tested it. With the features reduction Zhou, Han, Liu, He, and Wang (2018) recommended, his DL model had a 93% accuracy rate. Meidan et al. (2018) suggested a deep autoencoder and used the Mirai dataset to train the model. The model was then adjusted using various hyperparameters.

In 31, Kasongo et al. developed the FNN model and used the filter approach to choose the feature. Following that, the model was adjusted using a variety of hyperparameters and parameters, including the learning rate and the number of neurons in hidden layers. The model was evaluated and contrasted with other ML approaches using the NSL-KDD dataset. The author's model, which included three hidden layers with 30 neurons each and was tested on a binary classification dataset, had an accuracy of 88 percent. Similar results were obtained for multi-class classification utilizing 3 hidden layers and 150 neurons, which yielded an accuracy of 86.19

Almaiah et al. used the Frequency Particle Swarm Optimization (FPSO) approach in Almaiah and Almomani (2020) to identify the characteristics of the Shamoon attack. The Shamoon addresses industrial data, while the Fog nodes supply medical, educational, and industrial data. The source of the assault can be determined by locating the initial node because the author studied the Shamoon attack's movement and discovered that it follows the shortest path.

In Siam et al. (2021), Siam et al. proposed an IoT-based smart health monitoring system that uses sensors to evaluate temperature, blood oxygen levels, and heart rate. The Advanced Encryption Standard (AES) technique is then used to encrypt the data before it is delivered to the organization for decryption. A 95% confidence interval was achieved using the suggested procedure.

Almaiah et al. applied the blockchain concept for IIoT and proposed the deep learning model in Almaiah, Hajjej, Ali, Pasha, and Almomani (2022). The proposed model outperformed the current consensus protocol employed in the benchmark models, according to the results of the improvement in the blockchain's existing consensus protocol. In Ali et al. (2022), Ali et al. put out a blockchain-based model for the health care system and for the protection of data that uses a homomorphic encryption method. Using the Hyperledger Caliper, a hybrid DNN model with binary spring search (BSS) was put into practice for both intrusion detection and blockchain. The proposed approach obtained shorter confirmation time and computational cost for security compared to benchmark models.

Al Hwaitat et al. (2020) proposed the Particle Swarm Optimization (PSO) algorithm and compared the model to the existing optimization approaches. The program was improved to detect jamming attacks, which are the most prevalent kind of DoS attack. The outcome has shown that the suggested strategy produced superior outcomes in terms of the coverage area and the least fitness value. In Fatani et al. (2023), Fatani et al. proposed a deep learning model with an optimization technique. The author employed the CNN model for feature extraction, the growth optimizer modified version (MGO) for feature selection, and the whale optimization algorithm for the search process. An experiment using several datasets revealed that the MGO performed better than other strategies. The KDD dataset experiment shows that the training accuracy is 99.9 while the testing accuracy is 92.04. Similarly, the training accuracy on the NSL KDD dataset is 99.214. In contrast, the testing accuracy is 76.72, demonstrating that the model is over fit and performs well on the training dataset, but accuracy decreases on the testing dataset.

Similarly, in Abd Elaziz, Al-qaness, Dahou, Ibrahim, and Abd El-Latif (2023), the author builds the CNN-CapSA model for Intrusion detection using a combination of a deep learning model and the swarm intelligence method. The author used a deep learning model to find optimal features, and then an optimizer based on a swarm intelligence method called the Capuchin Search Algorithm (CapSA) was applied for efficient feature selection. The experiment was conducted on four different datasets.

2.2. Explainable AI and IDS studies based on model explanation

Explainable AI is being researched because ML and DL models are opaque and challenging to grasp, making it difficult to interpret model predictions. Explainable AI explains the predictions model, fostering model transparency and confidence. The idea of explainable AI is a new one, and it entails employing model explanation techniques to explain the models created and the contributions of each feature to the prediction (Ribeiro, Singh, & Guestrin, 2016; Samek, Wiegand, & Müller, 2017).

In Zhou, Hooker, and Wang (2021), Zhou et al. recommended stable LIME to be used for explaining models and deployed a random forest classifier to the data set containing breast cancer data, and it achieved 95% accuracy.

2.3. Local interpretable model-agnostic explanations (LIME)

LIME gives the user the model interpretation, which clarifies the forecast on a given instance. Having confidence in the model comes from understanding it, and LIME explains the predictions the model made. LIME checks the model using an instance's explanation as a basis. Eq. (1) provides the formula for LIME, which minimizes loss L and determines how closely the explanation resembles the original model. $\epsilon(x)$ is the explanation for instance x of the model g.

$$\epsilon(x) = argmin(L(f, g, \pi x)) + \Omega(g) \tag{1}$$

where, g represents the interpretable model $g\epsilon$ G. G represents the family of the model. πx is the proximity measure of the neighborhood we used to explain the instance. $\Omega(g)$ represents the complexity of the model eg, the number of features. f represents the probability of x belonging to specific class.

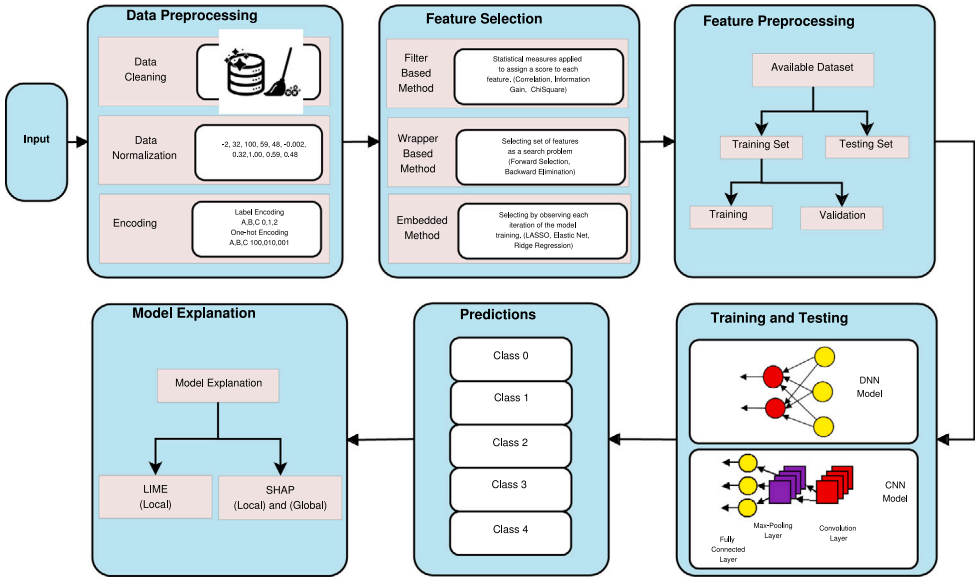


Fig. 1. Workflow of the proposed framework.

2.4. Shapley additive explanations (SHAP)

SHAP explains the model using Shapley values based on the feature importance. For a given data point for extracting the SHAP value in the Shapley value extraction formula, the contribution of the j feature is calculated using Eq. (2).

$$\phi(i) = \Sigma_{S \subseteq F/j}((|S|!(|P| - |S| - 1)!)/|P|!)(v(S \cup j) - v(S)) \quad (2)$$

where P is all features in the dataset and S is the set of all except the feature j, and $v(x)$ is the contribution of a subset x.

In summary, the review of the literature reveals that the researchers have proposed a variety of machine learning and deep learning techniques for intrusion detection and evaluated the model based on accuracy, precision, recall, and F1 score metrics using various benchmark datasets, but they have not provided any metrics for the model's trustworthiness and ability to explain the model. The feature selection also minimizes the number of features, which lowers the model's complexity. It also reduces the amount of time needed to test and train the model, which improves the model's performance. The model's accuracy and performance can be improved by integrating filter-based selection with deep learning methods. Since trustworthiness is not discussed in the literature, we reviewed our study's goal to make the IDS more trustworthy. The model must be trusted to be used in the real world. By utilizing the techniques of explainability, this study seeks to expand the field of XAI.

The literature review revealed that the described approaches' accuracy is high but that the DL and ML models are complex and that comprehension of the prediction is necessary in order to place trust in the model. Each feature contribution should be explained in the prediction. We can use the concept of explainable AI to explain each feature's prediction and contribution, which builds trust in the model. Secondly, the training time and complexity of the model should be reduced by reducing the features. Feature selection techniques could be used, which reduce the number of input variables to the model by eliminating redundant and irrelevant features and thus reducing both model training time and complexity.

3. Proposed framework

This section presents the workflow of the framework for detecting attacks in the network, as shown in Fig. 1. The main steps are Dataset Description, Data Preprocessing to encode and normalize the data, Feature Selection to identify important features and thus reduce the input variables, Feature Preprocessing to transform the dataset, and then training and testing the proposed deep learning method, and last explanation of the model. We explain them in detail as follows:

3.1. Dataset description

Using a network analyzer tool, raw traffic is gathered, and the features are then extracted. The researchers used publicly accessible datasets to test the DL models for intrusion detection systems. One of them NSL-KDD dataset is made up of 42 features in all, 38 of which are numerical values, 3 of which are nominal values, and one label indicates the normal/attack type category. Furthermore, another UNSW-NB15 dataset has 44 characteristics in total, including 4 categorical values, 39 numerical values, and one label indicating the category of normal/attack.

3.2. Data preprocessing

In this stage, the dataset is transformed and normalized after redundant data has been eliminated from it. During data transformation, various encoding techniques are used to translate the nominal values of the characteristics in the dataset into numerical values. Label encoding and one hot encoding are the two most popular encoding techniques. Data must be normalized such that values fall within the range of 0 to 1, enhancing the model's accuracy and performance. Min–Max normalization is used to normalize data (Sharma, Sharma, & Lal, 2022b).

3.3. Feature selection

Features are the input variables to the ML/DL models. The model is trained after selecting the important features, and the method is called the feature selection technique, which subsequently decreases the feature columns in the dataset, thus reducing storage and computing costs. Different feature selection approaches are:

3.3.1. Correlation-based filter method

The features are chosen using this method's correlation-based solution, where the correlation score and threshold value are used to determine which characteristics to use. Screening is used to remove duplicated features and highly associated characteristics from the dataset. Features with a correlation score higher than the threshold value are deemed highly correlated. The following step is dropping one of the highly associated attributes.

3.3.2. Wrapper methods

The induction methodology is used in this method to choose the feature subset. Through the use of backward elimination and forward selection techniques, the subset of features is chosen. Backward elimination involves starting with all of the set's features and then removing those that have a negative impact on the model's performance. When using forward selection, we begin with a subset that is empty and then add features to improve the model's performance and provide the optimal feature subset.

3.3.3. Embedded methods

To choose the best characteristics, this method combines the filter and wrapper methods. LASSO regularization and Random Forest are the two most used variations of this technique. In our experiment, we used filter-based method as the method requires less time and computation.

3.4. Feature preprocessing

After encoding and rescaling, the processed dataset is converted to model-compatible format, and the dataset is split into three parts, namely training and validation sets for training the dataset and testing sets for testing the model. The dataset is used to train the DNN model after normalization, encoding, and feature selection because it is now compatible with the model. The dataset must be transformed into 1D vector form before being applied to the 1D CNN model. To make the dataset compatible with the 2D CNN model, the processed data is converted to n×n matrix form, and the dataset is then applied to the model for training.

3.5. Training and testing the dataset

The training dataset is then applied to the deep learning training model and is classified as normal/attack class type, and validate the model with the validation dataset. DNN and CNN models are built during the training phase, and the training dataset is then fed as input. ~~CNN~~ model has a convolution layer, and max-pooling layers, and the DNN model has a dense layer. To prevent overfitting, the models are trained first and then tuned with various parameters and hyperparameters. After training, the deep learning model is tested with a testing dataset and classifies normal/attack categories.

3.6. Model explanation

After training and testing the model, we explain the prediction of the model. The concept of explainable AI is used to interpret the model prediction using a testing dataset. LIME and SHAP are two common surrogate models that aid in understanding the models. Model agnostic and modal specific are two broad categories of model explainability. Methods that are model-agnostic concentrate on input and output and can be used with any model, whereas a small number of models, such as linear regression, decision trees, and neural networks, are subjected to model-specific approaches. LIME provides the local explanation and explains the prediction of a particular instance. SHAP generates the explainer and provides local and global explanations. In a local explanation, a particular prediction is selected and explained using a plot based on features. Moreover, in a global explanation, the explanation is provided for the model predictions based on the features.

4. Evaluation and analysis

4.1. Dataset analysis

We used two publicly available datasets, namely NSL-KDD Cup and UNSW-NB15, to evaluate the model.

4.1.1. NSL-KDD dataset

KDDCup dataset was obtained from DARPA98 NIDS Evaluation Program managed by MIT Lincoln Labs, and the dataset is widely used in the field of NIDS. However, the main disadvantage of the dataset is that it contains duplicate or redundant records, so the new dataset named NSL-KDD was introduced that does not contain the duplicate or redundant records in the dataset. The dataset contains 42 features consisting of 3 nominal values, 38 numeric values, and 1 label showing the normal/attack category.

The dataset contains a total of 23 attack types, and we reduced the number of attack classes by grouping them into 4 main attack classes, namely Probe, DoS, U2R, and R2L.

The dataset consists of a total of 125 972 records having 67 342 records of normal class and 11 656, 995, 45 927, 52 records of Probe, R2L, DoS and U2R, respectively, as shown in Table 1 and the complete set of features are shown in Table 2.

4.1.2. UNSW-NB15 dataset

The lab of the Australian Centre for Cyber Security (ACCS) created the widely utilized dataset known as the UNSW-NB15 dataset to test the models. The dataset is made up of total of 44 features, where 39 are numerical values consisting of numbers, 4 nominal/categorical values made up of limited discrete values, and one label showing the normal/attack category. In the label feature column, there are ten classes with one normal class and nine different attack classes. The attack classes in the label column are Generic, Reconnaissance, DoS, Exploits, Analysis, Worms, Shellcode, Backdoor, and Fuzzers. The dataset contains 93 000 records of the Normal category and 164 673 attack categories. Total records in the dataset are 257 673, as shown in Table 4. The complete set of features is shown in Table 3. We selected five classes of normal and four attacks categories for experimental evaluation, namely Generic, Exploits, DoS, and Fuzzers. We selected a 50K sample size for class so that for classes having records greater than 50K, we selected a sample of size 50K, and for the class having records less than 50K, we selected all records. Normal and Generic classes have records greater than 50K size, so we randomly selected 50K size samples, and for the other three classes, namely fuzzers, DoS, and Exploits, records are less than 50K size, so we selected all records to resolve the class imbalance issue.

4.2. Data pre-processing

After the dataset extraction, we uploaded the CSV file format from the local drive on Google's Colaboratory and then imported the dataset into the pandas data frame. We dropped redundant columns and then encoded all the categorical features of the dataset into numeric values using the encoding method. One hot and label encoding are the two encoding methods available. In our experiment, we used label encoding, where each value in the column is converted into an integer. The dataset contains the values of different scales and needs to be normalized to a common scale ranging from 0 to 1.

NSL-KDD dataset contains 4 features which have categorical values "protocol_type", "flag", "services", "class" containing 3, 70, 11, 5 types, respectively. UNSW-NB15 contains 4 categorical value features, namely "proto", "service", "state", "attack_cat" containing 133, 13, 11, 10 types, respectively. We encoded using label encoding and converted the categorical values into integer values. The data normalization technique is applied to the dataset to convert the values of the dataset at a common scale. We used Min–max normalization, one of the several

Table 1
NSL-KDD CUP dataset attack statistics showing number of records in every normal/attack category.

Attack_class	Attack_SubType	No. of records
"Denial of Service (DoS)"	"apache2", "land", "pod", "smurf", "udpstorm", "worm" "back", "mailbomb", "teardrop", "neptune", "processtable"	45 927
"Probe"	"ipsweep", "mscan", "portsweep", "satan", "nmap", "saint"	11 656
Root to local (R2L)	"ftp_write", "httptunnel", "imap", "named", "phf", "sendmail", "Snmpgetattack", "snmpguess", "warezclient", "warezmaster", "xlock", "guess_passwd", "multihop", "spy", "xsnoop"	995
User to Root (U2R)	"buffer_overflow", "perl", "ps", "sqlattack", "xterm", "loadmodule", "rootkit"	52
Normal		67 342
Sum of records		**125 972**

Table 2
The features of NSL-KDD cup dataset.

S.No	Feature name	S.No	Feature name	S.No	Feature name
1	protocol_type	15	num_shells	29	srv_rerror_rate
2	src_bytes	16	num_access_files	30	root_shell
3	rerror_rate	17	serror_rate	31	dst_host_diff_srv_rate
4	is_guest_login	18	dst_host_serror_rate	32	num_root
5	srv_serror_rate	19	duration	33	dst_host_same_src_port_rate
6	diff_srv_rate	20	count	34	is_host_login
7	service	21	srv_count	35	dst_host_srv_serror_rate
8	num_failed_logins	22	wrong_fragment	36	num_file_creations
9	dst_host_count	23	dst_bytes	37	dst_host_srv_rerror_rate
10	num_compromised	24	land	38	num_outbound_cmds
11	dst_host_same_srv_rate	25	hot	39	dst_host_srv_count
12	su_attempted	26	urgent	40	same_srv_rate
13	Flag	27	srv_diff_host_rate	41	dst_host_srv_diff_host_rate
14	dst_host_rerror_rate	28	logged_in	42	class

Table 3
The features of UNSW-NB 15 dataset.

S.No	Feature name	S.No	Feature name	S.No	Feature name
1	Dur	16	djit	31	trans_depth
2	dpkts	17	ct_dst_src_ltm	32	response_body_len
3	spkts	18	ct_ftp_cmd	33	ct_srv_src
4	dbytes	19	ct_src_ltm	34	is_ftp_login
5	sbytes	20	is_sm_ips_ports	35	ct_dst_ltm
6	rate	21	swin	36	ct_dst_sport_ltm
7	Sttl	22	attack_cat	37	ct_src_dport_ltm
8	dttl	23	Stcpb	38	ct_state_ttl
9	sload	24	dtcpb	39	ct_flw_http_mthd
10	dload	25	dwin	40	ct_srv_dst
11	sloss	26	tcprtt	41	proto
12	dloss	27	synack	42	service
13	sinpkt	28	ackdat	43	state
14	dinpkt	29	smean	44	label
15	sjit	30	dmean		

Table 4
UNSW NB15 dataset attack statistics displaying number of records in every normal/attack category.

Attack category	No of records	Attack category	No of records
Generic	58 871	Exploits	44 525
DoS	16 353	Fuzzers	24 246
Analysis	2677	Reconnaissance	13 987
Backdoor	2329	Worms	174
Shellcode	1511	Total Attack	164 673
Normal			930 00
Total records			**257 673**

data normalization techniques available, to transform or normalize the dataset at a common scale as shown in Eq. (3)

$$F_{new} = (F - F_{min})/(F_{max} - F_{min}), \tag{3}$$

where F_{min} is the smallest value of the feature and F_{max} is the largest value of the feature, and the F is the actual value of the feature in the column. Furthermore, we get F_{new} as the new value, which lies in the range of 0 to 1. After label encoding, we then normalized the datasets using min–max normalization.

4.2.1. Feature selection

After transforming the dataset to the common scale, we select the important features by the feature selection techniques. We choose the correlation-based filter method to select the features from various feature selection techniques. Feature selection reduces the computational time and increases the storage efficiency. Using the correlation-based filter method, where the features are chosen based on correlation score, we picked the features. We applied the Pearson correlation method to find the association between the features. The similarity measure between two features/variables, F1 and F2, is given by Eq. (4) given below:

$$PCC = cov(F1, F2)/\sigma(F1)\sigma(F2), \tag{4}$$

where the covariance is denoted by cov, σ denotes the standard deviation, and the Pearson correlation coefficient is denoted by PCC, whose value ranges from −1 to 1. Highly correlated features have a PCC value near to −1 and 1, and uncorrelated features have a PCC value near 0. The features with high correlation values are considered redundant features; thus, depending on the threshold value, we reduce the redundancy by removing one of the features from the collection of highly redundant features. We chose a threshold of 0.95, and features with correlation values higher than this value was chosen and considered redundant. We then deleted one feature from the group of redundant features.

During model building, we applied correlation for feature selection. In NSL-KDD Cup dataset 6 features "srv_serror_rate", "dst_host_srv_rerror_rate", "num_root", "dst_host_serror_rate", "dst_host_srv_serror _ rate", "srv_rerror_rate" are dropped from the dataset. After dropping, the new dataset contains 36 features. Similarly we applied correlation in UNSW-NB15 and 'ct_src_dport_ltm', 'loss', 'dwin', 'ct_ftp_cmd', 'label', ' ct_srv_dst' features are dropped from the dataset and the new dataset

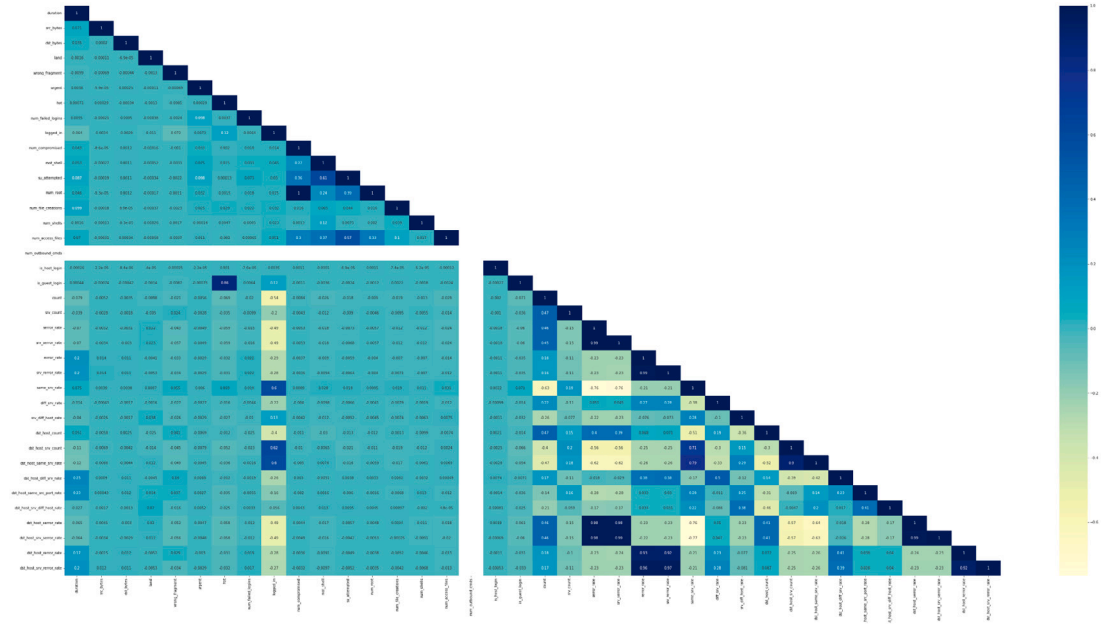


Fig. 2. Correlation matrix of the NSL-KDD dataset.

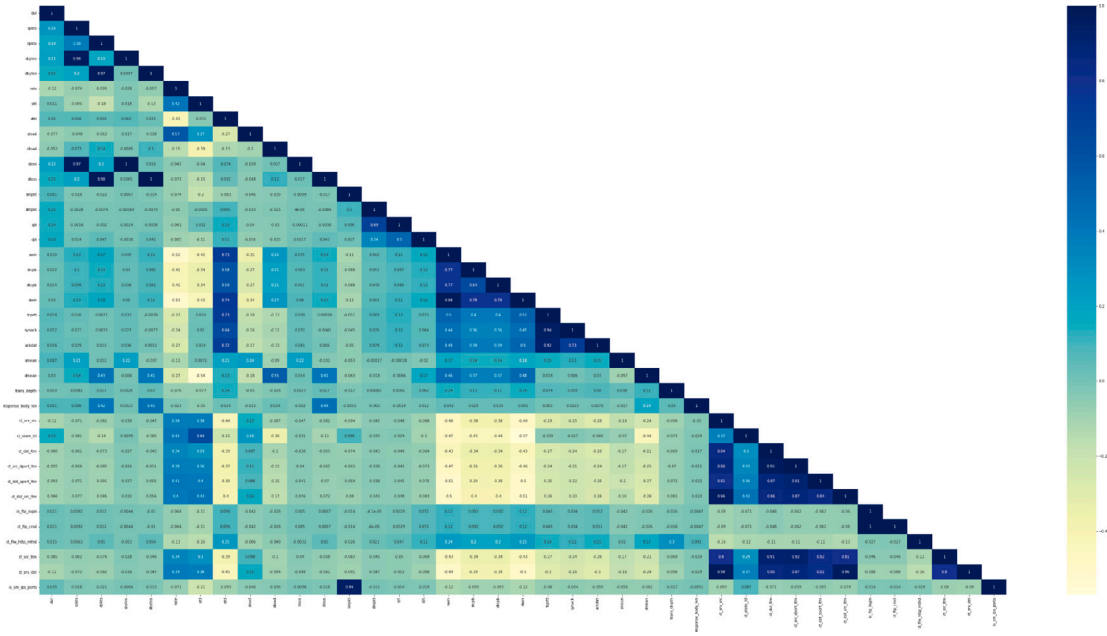


Fig. 3. Correlation matrix of the UNSW-NB 15 dataset.

contains 38 features. The correlation matrix of the features of the NSL-KDD and UNSW-NB15 datasets is shown in Figs. 2 and 3 respectively, which shows the highly correlated features. Features are chosen that have a value higher than the threshold of 0.95 and are regarded as redundant, which leads to the removal of redundant features by dropping one of the features. Fig. 4 shows the architecture of the DNN model representing the layers and neurons, while Fig. 5 presents the architecture of the CNN model representing a number of convolution, pooling layers with filter size. We presented the confusion matrix for classification in Fig. 6.

4.3. Feature preprocessing

The processed data is then made compatible with the model for training after feature selection. The datasets are applied as input to the DNN model and are compatible with it. However, the data is transformed into 1D vector form for the 1D CNN model, which is then used as input. The data is transformed into a 2D matrix for the 2D CNN model before being compatible with it.After encoding and scaling, the processed data is converted into 2D matrix form, which is then given as input into the CNN model. The dataset is transformed into a 2D matrix depending on the number of features. The dataset containing X features is rescaled into the $N \times N$ matrix, and if X is not a perfect square, then the record in the dataset is padded with '0' and then transformed into the nearest perfect square.

NSL-KDDnew dataset, after feature selection, has a total of 36 features ; therefore, in order to input the dataset into a 2D CNN model, we first transformed the features into a 6×6 matrix. Then, we input the dataset into the 2D CNN model, where each record of the dataset with its 36 features is transformed into a 6×6 pixels. The UNSW-NBnew

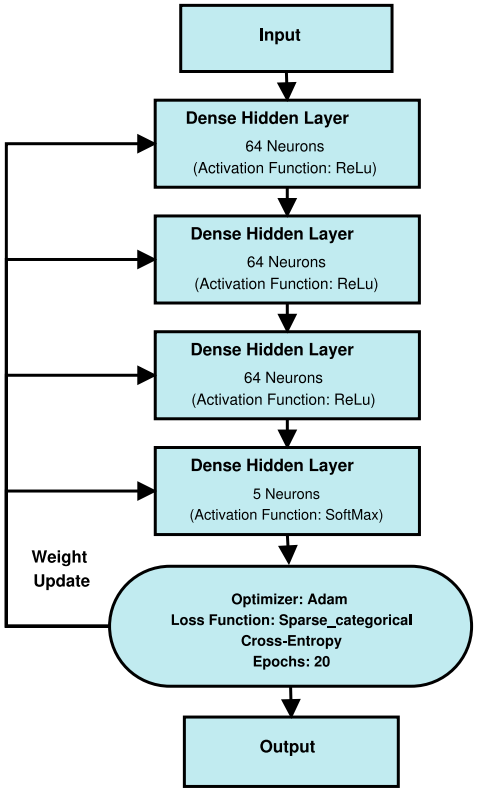


Fig. 4. Architecture of the DNN model.

dataset contains 38 features and is transformed into a 7×7 matrix with a padding of 11 pixels. We then split/divide the dataset into 25%–75% testing and training sets.

4.4. Training and testing the model

Following encoding, scaling, and transformation, the transformed, compatible dataset is used as the input for the training model, and the model is trained using the training dataset. We trained DNN and CNN models by randomly putting 75% rows in the training dataset and then further split into putting 60% rows for the training and 15% rows for the validation of the model. 25% dataset is applied for testing the model. DNN model contains dense hidden layers, and the CNN model contains three layers, namely a convolution layer, then a pooling layer, and at last, it contains a fully connected layer for classification. The convolution layer uses a filter or kernel, stride, padding, and an activation function is applied, and the output is termed a 'feature map'. Pooling is applied for dimensionality reduction of each feature map, such as Max pooling and Avg pooling. The resultant data from the convolution and pooling is then input to a fully connected layer. The model is trained using the training dataset during the training phase. In order to make sure that the training and testing accuracies are nearly comparable, the trained model is now tested in the testing phase using a new dataset called the testing dataset that contains normal/attack classes. The model is overfit if training accuracy is high and testing accuracy is low.

5. Experimental setup

We utilized the TensorFlow library to build the models after uploading the dataset to Google's Colaboratory. After feature selection and data preparation, the dataset is divided into three sets, each measuring 60%, 15%, and 25% of the total data set. These sets are referred to as the training set and validation set for training the model, and then the model trained is tested on the testing set.

Each of the three dense hidden layers in our DNN model, each with 64 neurons, contains five neurons in it. Dense hidden layers receive ReLu activation, but the last dense layer receives softmax activation, as seen in Fig. 4. Fig. 4 illustrates how we used the Adam optimizer. The sparse categorical loss function is used to optimize the model during training. The probability between the real value and predicted values is compared, and the loss, which measures how far the predicted value is from the actual value, is determined. Loss minimization is the goal during the model training. To prevent the model from being overfitting, we tweaked it using several hyperparameters like weight decay and dropout rate.

With a kernel size of (3, 3), ReLu activation function, and Max pooling with a pool size of (2, 2), we create a 2D-CNN model with three convolution layers and 64, 32, and 32 neurons. According to Fig. 5, the number of neurons in the last layer depends on the number of classes in the dataset, and there are 5 classes in the dataset; the dense layer, which is at the last of the model, has 5 neurons. The softmax activation function, which converts the n real values into values between 0 and 1, is utilized at the last layer. Additionally, we chose a 3 kernel size and a 2 pooling size for the 1D-CNN model. With the help of various hyperparameters, including kernel size and dropout rate, the model was tweaked.

To prevent overfitting, we trained the model with various hyperparameters, a weight decay of 0.0001, a dropout rate of 0, and a learning rate of 0.001 for epochs of 20 (Sharma, Sharma, & Lal, 2022a). Using a test dataset, we evaluated the model. For the NSL-KDDnew dataset, the accuracy obtained by the DNN, 1D-CNN, and 2D-CNN models is 0.993, 0.992, and 0.994, respectively. With the UNSW-NBnew dataset as our test subject, we were able to get accuracy scores of 0.80, 0.80, and 0.81 for the DNN, 1D-CNN, and 2D-CNN models, respectively.

Table 5
Model performance evaluation metrics.

Accuracy	A = (TrueP+TrueN)/(TrueP+TrueN+FalseP+FalseN)
Precision	P = TrueP/(TrueP+FalseP)
Recall	R = TrueP/(TrueP+FalseN)
F1 Score	F = 2PR/(P+R)

6. Result analysis

The performance of deep learning models is evaluated using different parameters. In our experiment, to evaluate the performance of the model, we have used the following evaluation metrics, where C1 and C2 are the two different classes:

- True-positives (TrueP): the outcome/predicted value which belongs to class C1 is accurately categorized as class C1.
- False-positive (FalseP): the outcome/predicted value which belongs to class C2 is incorrectly identified as class C1.
- True-negatives (TrueN): the outcome/predicted value which belongs to class C2 is accurately categorized as class C2.
- False-negative (FalseN): the outcome/predicted value which belongs to class C1 is incorrectly identified as class C2.

We presented the confusion matrix for classification in Fig. 6. We also calculated the following evaluation metrics as mentioned below and tabulated in Table 5

- Precision (P): It finds how accurate/ precise the model is by measuring the number of actual positives from the total number of predicted positives.
- Recall (R): It finds how many are predicted as positives out of the actual positives.

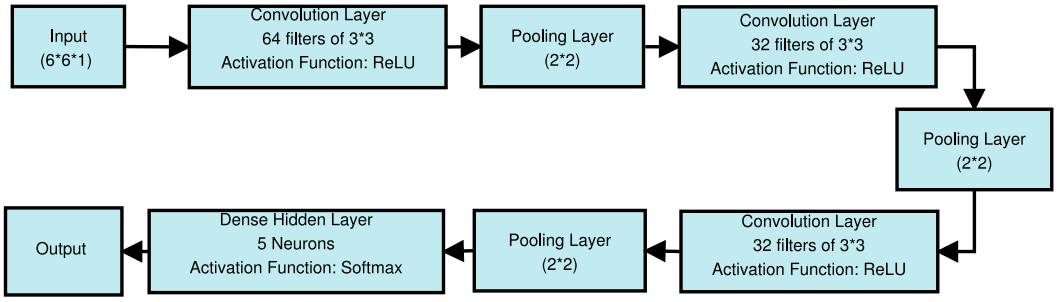


Fig. 5. Architecture of the CNN model.

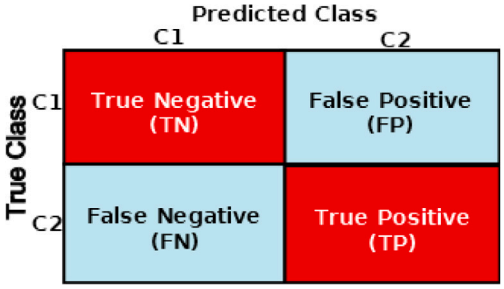


Fig. 6. Confusion matrix.

- F-measure (F1-Score): It finds the balance between Precision and Recall by calculating the harmonic mean of recall (R) and precision (P).
- Accuracy (ACC): It finds how many are correctly predicted as positives and negatives from the total number of predictions made.

6.1. Analysis and comparison of results

After training the DNN and CNN model for 20 epochs, we evaluate the performance of the model by testing the model on unseen data. We verified the model with testing data of both datasets, namely NSL-KDD and UNSW-NB15, for multi-class classification. The evaluation metrics used for evaluating the performance of the model are as follows:

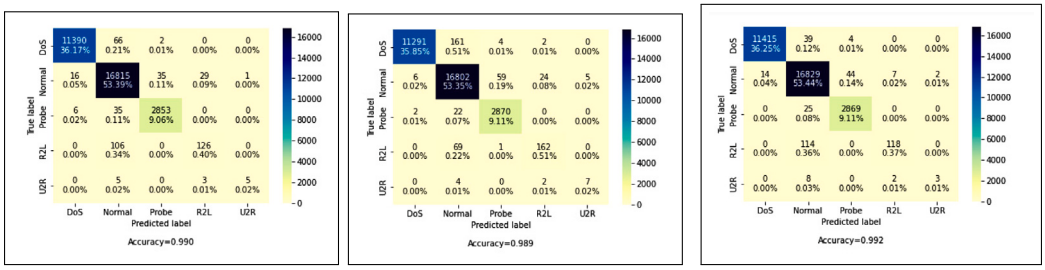
1. Confusion matrix: It presents the different outcomes in the table form and helps to visualize the outcome of the model. We plot the confusion matrix for both datasets containing the number and percentage of different normal/attack classes. The number of True Positives (TP) for each class is derived from the diagonal elements in the confusion matrix. The DNN, 1D-CNN, and 2D-CNN model's confusion matrix for the testing set of the NSL-KDDnew dataset is shown in Fig. 7, and for UNSW-NBnew dataset is shown in Fig. 9.
2. Precision, recall, and F1-Score: we derived the value of P, R, and F1-Score for the NSL-KDDnew dataset as shown in Table 6 and for UNSW-NBnew as shown in Table 7. High precision shows that the False Positives (FP) are less, and the high recall value shows that the False Negatives (FN) are less. Comparison of P, R, and F1-Score evaluation metrics of deep learning models for the NSL-KDDnew dataset is shown in Fig. 8, and for UNSW-NBnew dataset is shown in Fig. 10.
3. Accuracy and Loss: The NSL-KDDnew dataset: The proposed DNN model with a 0.001 learning rate, 0.0001 weight decay, 0.01 dropout rate, and 20 epochs achieved the accuracy of 0.99 and loss is 0.04. The 2D CNN model with a 0.001 learning rate is trained to 20 epochs and attained the training and testing accuracy of 0.994, with a loss of 0.01. 1D-CNN model achieved an accuracy of 0.989 and a loss is 0.02. A model with higher accuracy and reduced loss is considered better. The results show that the 2D CNN model performs better in accuracy and loss, as shown in Figs. 11(a) and 11(b).
   The UNSW-NBnew dataset: The proposed DNN model with the same parameters attained an accuracy of 0.80 and a loss of 0.47. 1D-CNN model with the same kernel and pool size achieved an accuracy of 0.80 and a loss of 0.41. 2D-CNN model with the same configuration attained an accuracy of 0.81 and a loss of 0.40, as shown in Figs. 11(c) and 11(d).
4. Model Training Time: We evaluated the training time of the model. The difference between the start and end times is evaluated during model training. The training times for the DNN, 1D CNN, and 2D CNN models for the NSL-KDDnew dataset are 142 ms, 325 ms, and 340 ms, respectively. The training times for DNN, 1DCNN, and 2DCNN models are 323 ms, 442 ms, and 455 ms for the UNSW-NBnew dataset, respectively. The 2D-CNN model achieved higher accuracy for both datasets; however, the time taken to train the CNN model is more than the DNN model. The DNN model trained faster on two separate datasets.

We also compared the proposed Deep learning models with the other machine learning (ML) models such as "k-nearest neighbors algorithm (KNN)", "Decision Tree (DT)", and "support vector machine (SVM)" models using the accuracy evaluation metric for multi-class classification as shown in Fig. 12. We applied the same label encoding, normalization methods, and selected features using correlation, and trained the models using two datasets, NSL-KDDnew, and UNSW-NBnew datasets. We implemented ML and DL techniques and achieved higher accuracy using DL models with less training time. We conclude that the 2D-CNN model achieved the highest accuracy for both datasets, but the training time of 2D-CNN is more than the DNN model.

6.2. Model explanation

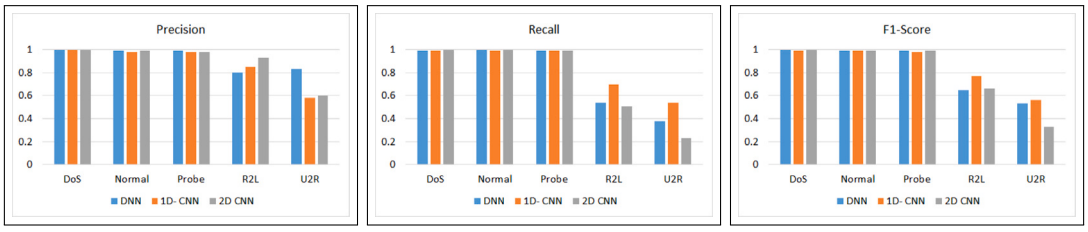
The DNN and 2D CNN models outperformed the other models when we compared them on two separate datasets. We also discovered that the DNN model required less time for model training when comparing the training times of the CNN model. Therefore, we chose the DNN model, and to build trust, we used explain ability tools to the DNN model.

- **LIME Model Explanation:** Select a particular instance in the testing dataset to get the probability values for each class. The LIME method explains the reason for assigning the probability to each class. Probability values are compared with the actual class of the instance.
  Instance prediction: Depending upon the values and weight assigned to the features, class probability is calculated, and then predict the class (Sharma, Sharma, & Lal, 2023). We selected four instances from the testing set and predicted the probability values of each class, as shown in Figs. 13 and 14. The left side of the Figure shows the probability of each class, and in the middle bar



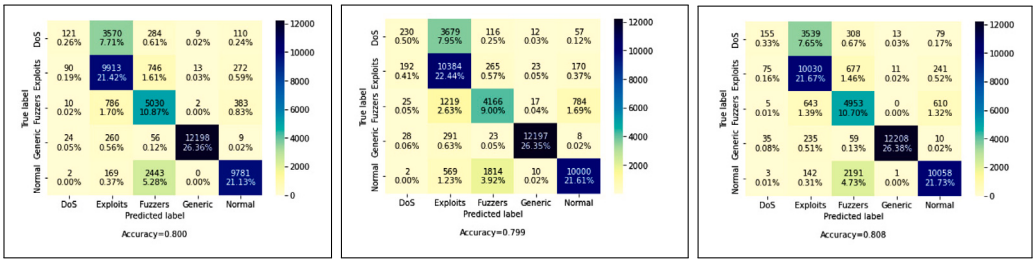
(a) Mouse (b) Gull (c) Tiger

Fig. 7. Confusion matrix of (a) DNN, (b) 1D CNN and (c) 2D CNN model for NSL-KDDnew dataset.



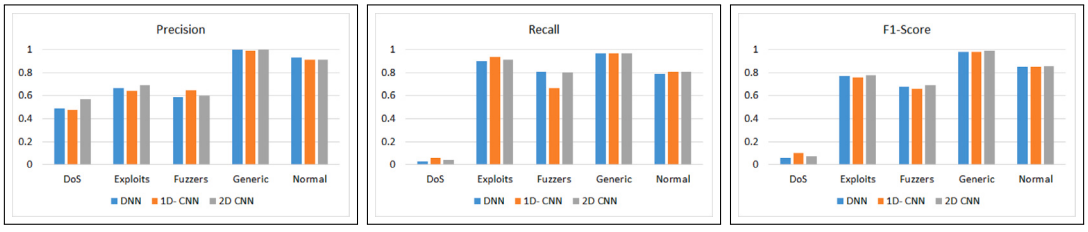
(a) Precision (b) Recall (c) F1-score

Fig. 8. (a) Precision, (b) Recall and (c) F1-Score of DNN, 1D-CNN and 2D-CNN model for NSL-KDDnew.



(a) Mouse (b) Gull (c) Tiger

Fig. 9. Confusion matrix of DNN, 1D CNN and 2D CNN model for UNSW-NBnew.



(a) Mouse (b) Gull (c) Tiger

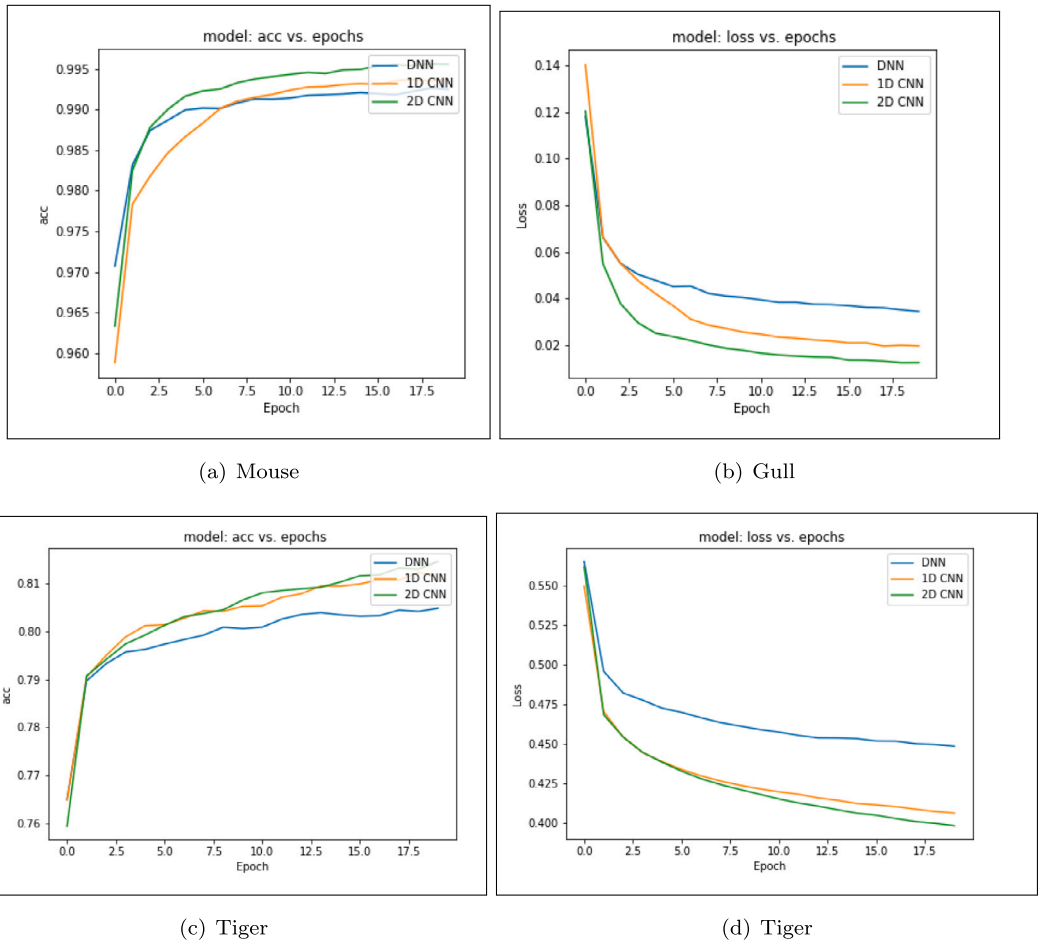
Fig. 10. (a) Precision, (b) Recall and (c) F1-Score of DNN, 1D-CNN and 2D-CNN model for UNSW-NBnew.

Table 6
Evaluation metrics of DNN, 1D-CNN and 2D-CNN model for NSL-KDDnew.

Classification report									
Attacks	DNN model			ID CNN model			2D CNN model		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
DoS	1.00	0.99	1.00	1.00	0.99	0.99	1.00	1.00	1.00
Normal	0.99	1.00	0.99	0.98	0.99	0.99	0.99	1.00	0.99
Probe	0.99	0.99	0.99	0.98	0.99	0.98	0.98	0.99	0.99
R2L	0.80	0.54	0.06	0.85	0.70	0.77	0.93	0.51	0.66
U2R	0.83	0.38	0.53	0.58	0.54	0.56	0.60	0.23	0.33
Accuracy			**0.99**			**0.99**			**0.99**

Table 7
Evaluation metrics of DNN, 1D-CNN and 2D-CNN model for UNSW-NBnew dataset.

Classification report									
Attacks	DNN model			ID CNN model			2D CNN model		
	Precision	Recall	F1-Score	Precision	Recall	F1-Score	Precision	Recall	F1-Score
DoS	0.49	0.03	0.06	0.48	0.06	0.10	0.57	0.04	0.07
Exploits	0.67	0.90	0.77	0.64	0.94	0.76	0.69	0.91	0.78
Fuzzers	0.59	0.81	0.68	0.65	0.67	0.66	0.60	0.80	0.69
Generic	1.00	0.97	0.98	0.99	0.97	0.98	1.00	0.97	0.99
Normal	0.93	0.79	0.85	0.91	0.81	0.85	0.91	0.81	0.86
Accuracy			**0.80**			**0.80**			**0.81**



(a) Mouse (b) Gull

(c) Tiger (d) Tiger

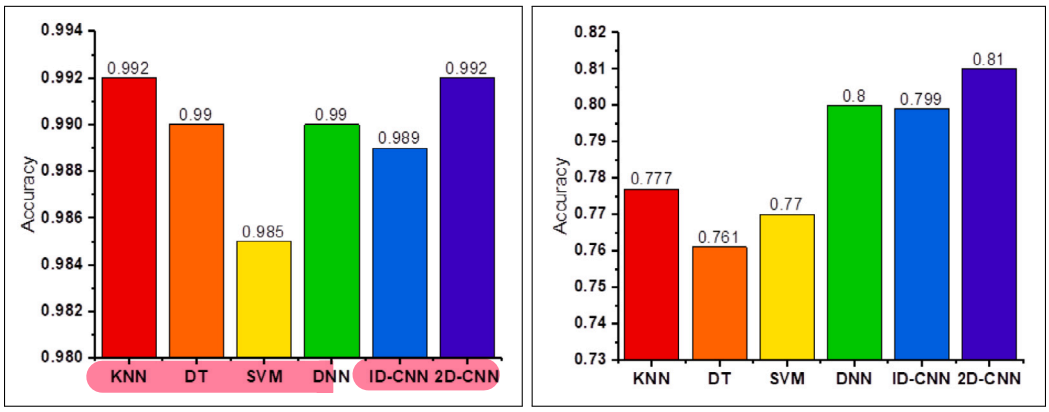
Fig. 11. (a) Accuracy vs. Epochs for NSL-KDDnew dataset (b) Loss vs. Epochs for NSL-KDDnew dataset (c) Accuracy vs. Epochs for UNSW-NBnew dataset (d) Loss vs. Epochs of UNSW-NBnew dataset.

chart, it shows the features of interest, and in the right, it shows the value of features. Orange specifies the positive impact, and blue specifies the negative impact of the feature.

Fig. 13(a) shows that the actual value is Normal, and the predicted value is normal. On the left, we see that the model predicted as Normal with 100% accuracy, and in the middle, it shows the top ten features. The right side of the bar chart shows the features which help to predict the instance as Normal, and the left side shows the features which help to predict the instance as not Normal. To predict instance as Normal, the features wrong_fragment, hot, serror_rate, rerror_rate, su_attempted have values ≤ 0.00, and the weight assigned are 0.40, 0.39, 0.30, 0.23, 0.16 respectively. And to predict an instance as Not Normal, the features protocol_type and num_shells have values ≤ 0.50 and 0.00, respectively and the weights assigned are 0.14 and 0.09. The value of features is shown on the right side for the selected instance. Logged_in has a value of 1.00, and the weight assigned in the bar chart is 0.26, the value of protocol_type is 0.5, and the weight assigned is 0.14. The total value is 0.26 for Normal and 0.07 for Not Normal, and since the value of Normal is greater than Not Normal, the model predicted the instance as Normal. Similarly, the actual value is DoS, and the predicted value is DoS with 100% accuracy, as shown in Fig. 13(b).

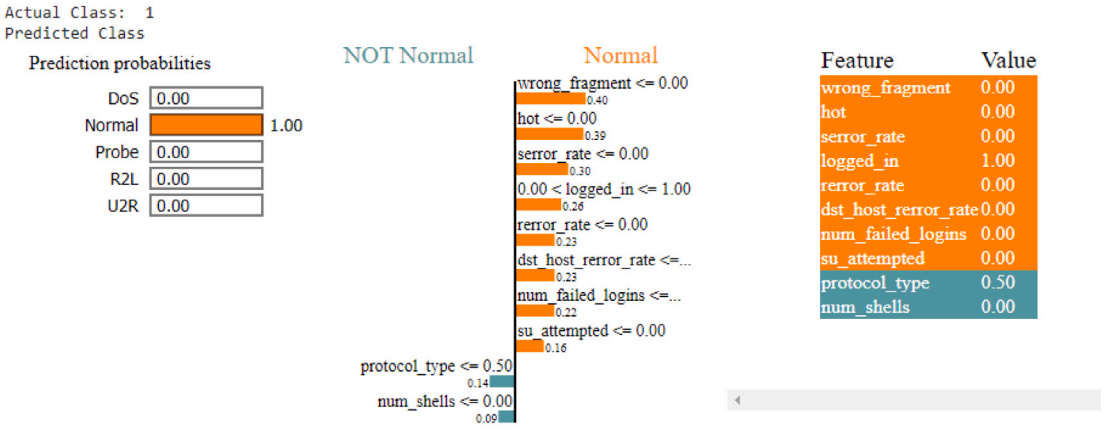
For UNSW-NB 15, the actual value is Normal, and the predicted value is Normal with 100% accuracy, as shown in Fig. 14(a). Another instance selected to form the testing set shows that the actual value is Exploits and the predicted value is Exploits with 51% accuracy, as shown in Fig. 14(b).

- **SHAP Model Explanation:** SHAP is used widely for explaining models and understanding how the features are related to the predictions. SHAP provides the local and global explanation. In local explanation, we select a particular instance and explain

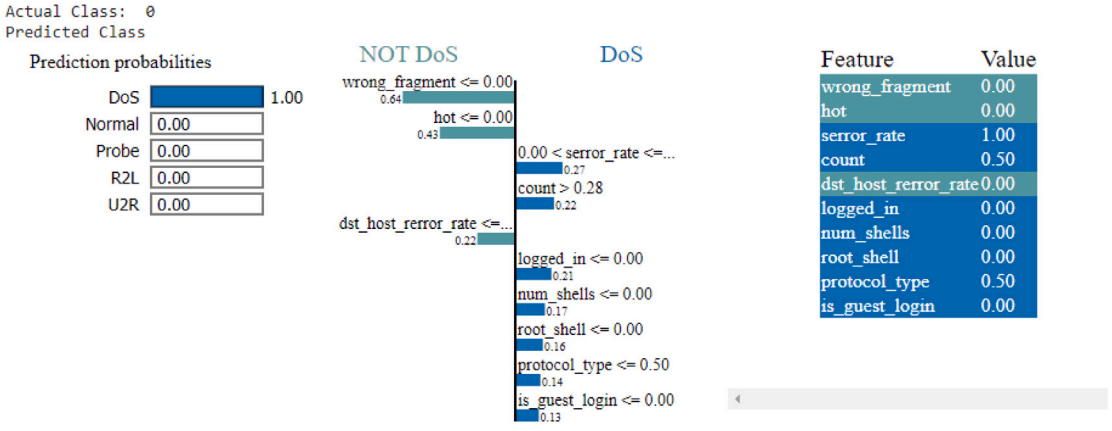


(a) KDD-CUP (b) UNSW-NB

Fig. 12. Comparisons of accuracy of different models.



(a) Mouse

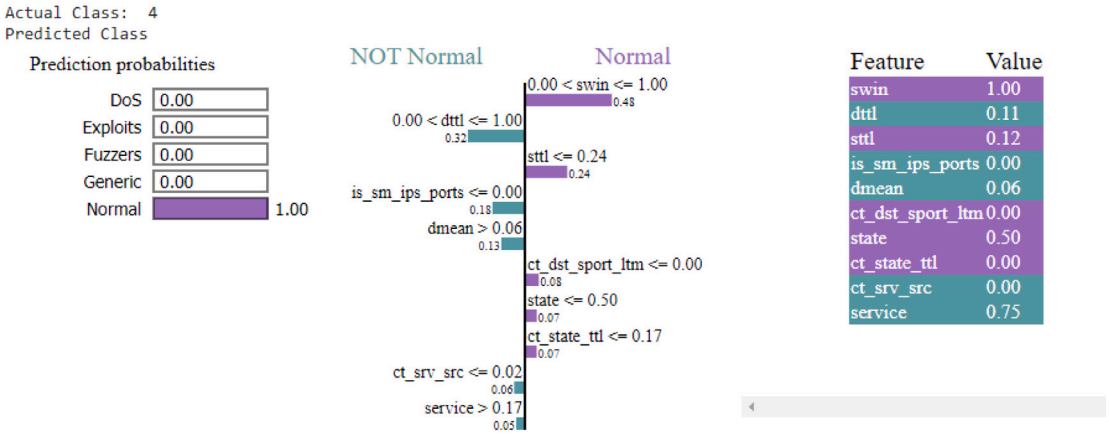


(b) Gull

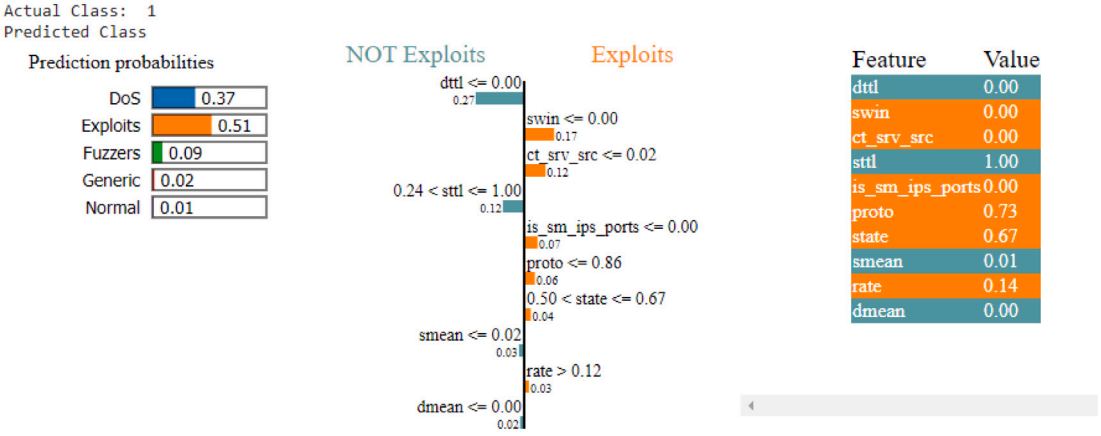
Fig. 13. LIME explanation of NSL-KDDnew dataset instances.

the model prediction showing each feature's contribution to the prediction of the instance selected. In the global explanation, we explain the model prediction using the contribution of each feature in the prediction.

SHAP calculates the Shapley value, which shows the impact of features on the model predictions. We selected a particular instance and calculated the shape values. Fig. 15 shows the local plot of the DoS instance, showing each feature's contribution to the prediction. The plot shows the base value, and the features having a positive impact on the prediction are in red, and the features showing a negative impact on the predictions are in blue. The base value in the plot is the average of all prediction values. Each strip in the plot shows the impact of the features in pushing the predicted value close or farther from the base value.



(a) Mouse



(b) Gull

Fig. 14. LIME explanation of UNSW-NBnew dataset instances.

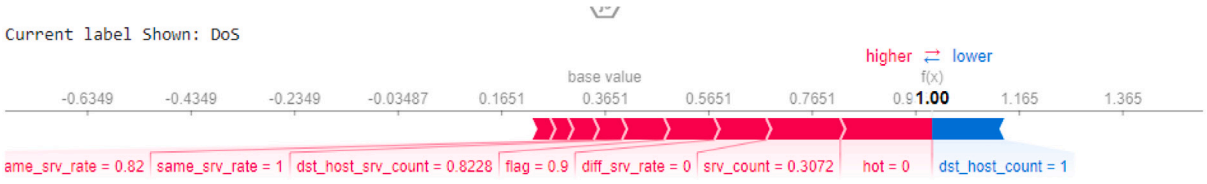


Fig. 15. The local plot of DoS instance of NSL-KDDnew showing the each feature's contribution to the prediction.

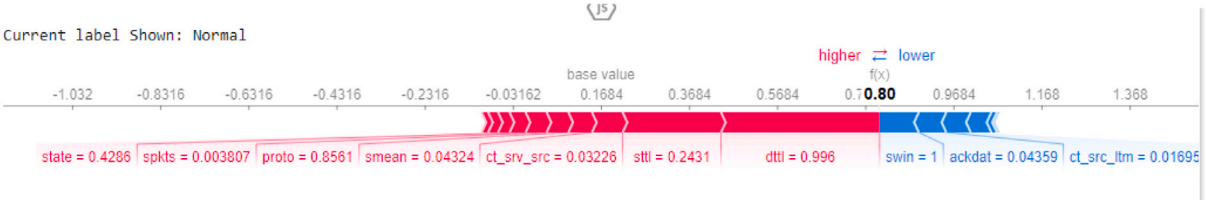


Fig. 16. The local plot of normal instance of UNSW-NBnew showing the each feature's contribution to the prediction.

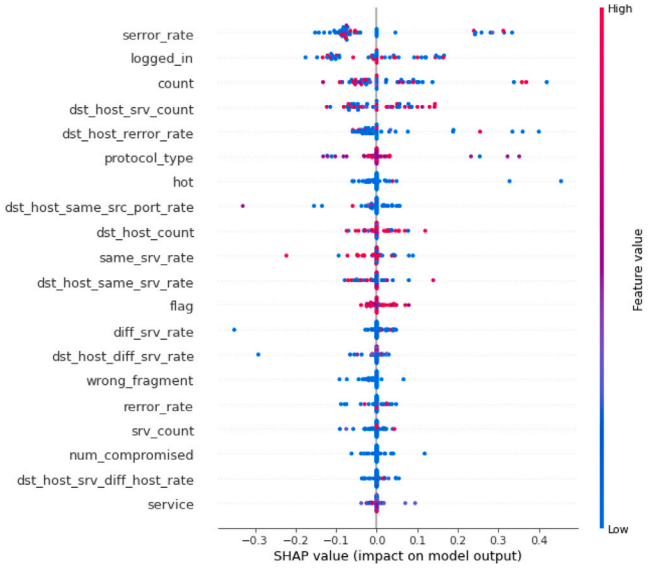


Fig. 17. Summary plot of DoS class of NSL-KDDnew dataset.

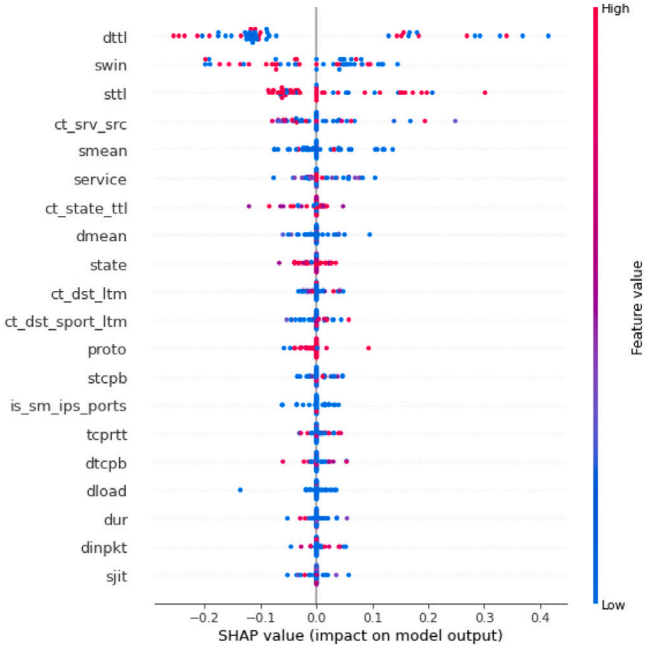


Fig. 18. Summary plot of normal class of UNSW-NBnew.

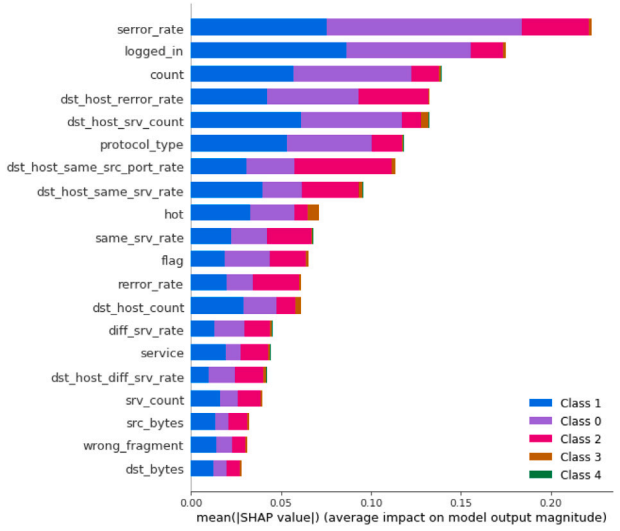


Fig. 19. Force plot of NSL-KDDnew dataset.

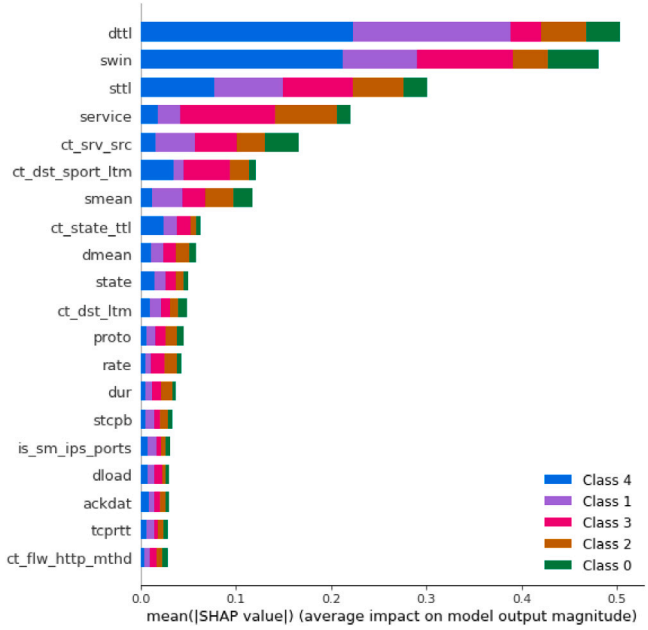


Fig. 20. Force plot of UNSW-NBnew.

Red strip features push the value to higher values, whereas blue strip features push the value to lower values. The contribution of features having wider strips is more.

The base value is 0.361, and for the particular record selected from the NSL-KDDnew dataset, the features ‘same_srv_rate’, ‘dst_host_same_srv_rate’, ‘dst_host_srv_count’, ‘flag’, ‘diff_srv_rate, srv_count’’, srv_count’ and hot have a positive contribution on the prediction value, and dst_host_count have a negative contribution. Hot is the most important feature as the contribution has a wider range. The total positive contribution is greater than the negative contribution, and the final predicted value is greater than the base value, so the predicted class is DoS.

Similarly, we selected the record from the UNSW-NBnew dataset and found that the base value is 0.1684, and the predicted value is 0.80, as shown in Fig. 16. The feature ’dttl’ has a wider range and is the most important feature. For a global explanation, we plot a summary chart. We selected 50 samples of the testing dataset and derived a summary plot for both datasets where each point in the row shows the sharp value of testing samples. The higher value impacts positively on the prediction, and the lower value contributes negatively. Serror_rate is the most important feature for the DoS class in NSL-KDD dataset, as shown in Fig. 17. The most important feature for the Normal class in the UNSW-NB dataset is ‘data’, as shown in Fig. 18.

Force plot shows that for the NSL-KDD dataset, ‘serror_rate’ is the most important feature, and class labels are 0 for DoS, 1 for Normal, 2 for Probe, 3 for R2L, 4 for U2R in the NSL-KDDnew datasets, as shown in Fig. 19. Similarly, the Force plot shows that ‘dttl’ is the most important feature, and class labels are 0 for DoS, 1 for Exploits, 2 for Fuzzers, 3 for Generic, 4 for Normal classes in the UNSW-NBnew datasets, as shown in Fig. 20.

7. Conclusions

In this manuscript, we study different layers of IoT with the main security issues in each layer and propose a deep learning model for intrusion detection, and explain the model using an explainable AI concept to build trust in the model. In multiclass classification, our Deep learning model achieved higher accuracy for both datasets. We also explained the DNN model using LIME and SHAP. The dataset applied to the model is not balanced, so the accuracy of the class having a majority number of records are high as compared to the number of minority class, so our future work is to resolve the issue and improve the accuracy of minority classes in the dataset. Our proposed method achieved high accuracy with reduced training time. However, several issues still need to be improved.

The proposed model can be applied to other datasets, and the predictions of the model can be explained using LIME and SHAP methods. We reduced the number of features by selecting fewer features, which reduced the number of inputs and decreased the computational cost. Our future work is to apply other feature reduction techniques and find the optimal number of features that gives high accuracy to the model. The limitation of our model is the class imbalance issue in the dataset. We need to resolve the class imbalance issue by generating the synthetic data using GANs, applying other feature reduction techniques, and finding the best possible set of features. We applied LIME and SHAP to explain the model only, but after visualizing the outcome of SHAP and LIME, we can make the changes to the DNN model and find the best model for the dataset.

CRediT authorship contribution statement

Bhawana Sharma: Conceptualization, Methodology, Software, Writing – original draft. **Lokesh Sharma:** Visualization, Investigation, Supervision. **Chhagan Lal:** Supervision, Software. **Satyabrata Roy:** Writing – review & editing, Supervision, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request

References

Abd Elaziz, M., Al-qaness, M. A., Dahou, A., Ibrahim, R. A., & Abd El-Latif, A. A. (2023). Intrusion detection approach for cloud and IoT environments using deep learning and Capuchin search algorithm. *Advances in Engineering Software*, *176*, Article 103402.

Ahmad, Z., Shahid Khan, A., Wai Shiang, C., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32*(1), Article e4150.

Al-Fuqaha, A., Guizani, M., Mohammadi, M., Aledhari, M., & Ayyash, M. (2015). Internet of things: A survey on enabling technologies, protocols, and applications. *IEEE Communications Surveys & Tutorials*, *17*(4), 2347–2376.

Al-Garadi, M. A., Mohamed, A., Al-Ali, A. K., Du, X., Ali, I., & Guizani, M. (2020). A survey of machine and deep learning methods for internet of things (IoT) security. *IEEE Communications Surveys & Tutorials*, *22*(3), 1646–1685.

Al Hwaitat, A. K., Almaiah, M. A., Almomani, O., Al-Zahrani, M., Al-Sayed, R. M., Asaifi, R. M., et al. (2020). Improved security particle swarm optimization (PSO) algorithm to detect radio jamming attacks in mobile networks. *International Journal of Advanced Computer Science and Applications*, *11*(4).

Al Nafea, R., & Almaiah, M. A. (2021). Cyber security threats in cloud: Literature review. In *2021 international conference on information technology (ICIT)* (pp. 779–786). IEEE.

Ali, A., Almaiah, M. A., Hajjej, F., Pasha, M. F., Fang, O. H., Khan, R., et al. (2022). An industrial IoT-based blockchain-enabled secure searchable encryption approach for healthcare systems using neural network. *Sensors*, *22*(2), 572.

Almaiah, A., & Almomani, O. (2020). An investigation of digital forensics for shamoon attack behaviour in FOG computing and threat intelligence for incident response. *Journal of Theoretical and Applied Information Technology*, *15*, 98.

Almaiah, M. A., Hajjej, F., Ali, A., Pasha, M. F., & Almomani, O. (2022). A novel hybrid trustworthy decentralized authentication and data preservation model for digital healthcare IoT based CPS. *Sensors*, *22*(4), 1448.

Altulaihan, E., Almaiah, M. A., & Aljughaiman, A. (2022). Cybersecurity threats, countermeasures and mitigation techniques on the IoT: Future research directions. *Electronics*, *11*(20), 3330.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., & Faruki, P. (2019). Network intrusion detection for IoT security based on learning techniques. *IEEE Communications Surveys & Tutorials*, *21*(3), 2671–2701.

Da Xu, L., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on Industrial Informatics*, *10*(4), 2233–2243.

Fatani, A., Dahou, A., Abd Elaziz, M., Al-Qaness, M. A., Lu, S., Alfadhli, S. A., et al. (2023). Enhancing intrusion detection systems for IoT and cloud environments using a growth optimizer algorithm and conventional neural networks. *Sensors*, *23*(9), 4430.

Fenanir, S., Semchedine, F., & Baadache, A. (2019). A machine learning-based lightweight intrusion detection system for the internet of things. *Revista d'Intelligence Artificial*, *33*(3), 203–211.

Ge, M., Syed, N. F., Fu, X., Baig, Z., & Robles-Kelly, A. (2021). Towards a deep learning-driven intrusion detection approach for Internet of Things. *Computer Networks*, *186*, Article 107784.

Hassan, M. M., Gumaei, A., Alsanad, A., Alrubaian, M., & Fortino, G. (2020). A hybrid deep learning model for efficient intrusion detection in big data environment. *Information Sciences*, *513*, 386–396.

Karatas, G., Demir, O., & Sahingoz, O. K. (2020). Increasing the performance of machine learning-based IDSs on an imbalanced and up-to-date dataset. *IEEE Access*, *8*, 32150–32162.

Kasongo, S. M., & Sun, Y. (2020). A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Computers & Security*, *92*, Article 101752.

Khan, Z. A., & Herrmann, P. (2019). Recent advancements in intrusion detection systems for the internet of things. *Security and Communication Networks*, *2019*.

Kim, J., Kim, J., Kim, H., Shim, M., & Choi, E. (2020). CNN-based network intrusion detection against denial-of-service attacks. *Electronics*, *9*(6), 916.

Liang, C., Shanmugam, B., Azam, S., Jonkman, M., De Boer, F., & Narayansamy, G. (2019). Intrusion detection system for Internet of Things based on a machine learning approach. In *2019 international conference on vision towards emerging trends in communication and networking (ViTECoN)* (pp. 1–6). IEEE.

Lin, J., Yu, W., Zhang, N., Yang, X., Zhang, H., & Zhao, W. (2017). A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications. *IEEE Internet of Things Journal*, *4*(5), 1125–1142.

Ma, W. (2020). Analysis of anomaly detection method for internet of things based on deep learning. *Transactions on Emerging Telecommunications Technologies*, *31*(12), Article e3893.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Shabtai, A., Breitenbacher, D., et al. (2018). N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, *17*(3), 12–22.

Nagisetty, A., & Gupta, G. P. (2019). Framework for detection of malicious activities in IoT networks using keras deep learning library. In *2019 3rd international conference on computing methodologies and communication (ICCMC)* (pp. 633–637). IEEE.

Qiu, H., Dong, T., Zhang, T., Lu, J., Memmi, G., & Qiu, M. (2020). Adversarial attacks against network intrusion detection in IoT systems. *IEEE Internet of Things Journal*, *8*(13), 10327–10335.

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1135–1144).

Samek, W., Wiegand, T., & Müller, K.-R. (2017). Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296.

Sharma, B., Sharma, L., & Lal, C. (2019). Anomaly detection techniques using deep learning in IoT: a survey. In *2019 international conference on computational intelligence and knowledge economy (ICCIKE)* (pp. 146–149). IEEE.

Sharma, B., Sharma, L., & Lal, C. (2022a). Anomaly based network intrusion detection for IoT attacks using convolution neural network. In *2022 IEEE 7th international conference for convergence in technology (I2CT)* (pp. 1–6). http://dx.doi.org/10.1109/I2CT54291.2022.9824229.

Sharma, B., Sharma, L., & Lal, C. (2022b). Feature selection and deep learning technique for intrusion detection system in IoT. In *Proceedings of international conference on computational intelligence: ICCI 2020* (pp. 253–261). Springer.

Sharma, B., Sharma, L., & Lal, C. (2023). Anomaly-based DNN model for intrusion detection in IoT and model explanation: Explainable artificial intelligence. In *Proceedings of second international conference on computational electronics for wireless communications: ICCWC 2022* (pp. 315–324). Springer.

Siam, A. I., Almaiah, M. A., Al-Zahrani, A., Elazm, A. A., El Banby, G. M., El-Shafai, W., et al. (2021). Secure health monitoring communication systems based on IoT and cloud computing for medical emergency applications. *Computational Intelligence and Neuroscience*, *2021*.

Sun, P., Liu, P., Li, Q., Liu, C., Lu, X., Hao, R., et al. (2020). DL-IDS: extracting features using CNN-LSTM hybrid network for intrusion detection system. *Security and Communication Networks*, *2020*.

Teng, S., Wu, N., Zhu, H., Teng, L., & Zhang, W. (2017). SVM-DT-based adaptive and collaborative intrusion detection. *IEEE/CAA Journal of Automatica Sinica*, *5*(1), 108–118.

Thamilarasu, G., & Chawla, S. (2019). Towards deep-learning-driven intrusion detection for the internet of things. *Sensors*, *19*(9), 1977.

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

Vinayakumar, R., Alazab, M., Srinivasan, S., Pham, Q.-V., Padannayil, S. K., & Simran, K. (2020). A visualized botnet detection system based deep learning for the internet of things networks of smart cities. *IEEE Transactions on Industry Applications*, *56*(4), 4436–4456.

Xiao, Y., Xing, C., Zhang, T., & Zhao, Z. (2019). An intrusion detection model based on feature reduction and convolutional neural networks. *IEEE Access*, *7*, 42210–42219.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., et al. (2018). Machine learning and deep learning methods for cybersecurity. *Ieee Access*, *6*, 35365–35381.

Xu, H., Fang, C., Cao, Q., Fu, C., Yan, L., & Wei, S. (2018). Application of a distance-weighted KNN algorithm improved by moth-flame optimization in network intrusion detection. In *2018 IEEE 4th international symposium on wireless systems within the international conferences on intelligent data acquisition and advanced computing systems (IDAACS-SWS)* (pp. 166–170). IEEE.

Zhou, Y., Han, M., Liu, L., He, J. S., & Wang, Y. (2018). Deep learning approach for cyberattack detection. In *IEEE INFOCOM 2018-IEEE conference on computer communications workshops (INFOCOM WKSHPS)* (pp. 262–267). IEEE.

Zhou, Z., Hooker, G., & Wang, F. (2021). S-lime: Stabilized-lime for model explanation. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 2429–2438).