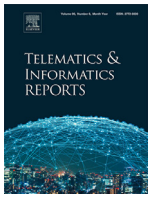
Telematics and Informatics Reports 11 (2023) 100077

Contents lists available at ScienceDirect

Telematics and Informatics Reports

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



Dependable intrusion detection system using deep convolutional neural network: A Novel framework and performance evaluation approach

Vanlalruata Hnamte*, Jamal Hussain

Department of Mathematics and Computer Science, Mizoram University, Tanhril, Aizawl, 796004, Mizoram, India

ARTICLE INFO

Keywords:
ISCX
DCNN
CICIDS2017
CICIDS2018
Kaggle DDoS

ABSTRACT

Intrusion detection systems (IDS) play a critical role in safeguarding computer networks against unauthorized access and malicious activities. However, traditional IDS approaches face challenges in accurately detecting complex and evolving cyber threats. The proposed framework leverages the power of deep learning to automatically extract meaningful features from network traffic data, enabling more accurate and robust intrusion detection. The proposed deep convolutional neural network (DCNN) has been trained on large-scale datasets, incorporating both normal and malicious network traffic, to enable effective discrimination between normal and anomalous behavior. To evaluate the performance of the framework, a comprehensive performance evaluation approach is developed, considering key metrics such as detection accuracy, false positive rate, and computational efficiency. Additionally, GPU has been utilized for boosting the performance of the model, demonstrating the effectiveness and superiority of the deep CNN-based intrusion detection system over traditional methods. The novelty of this study lies in the development of a dependable intrusion detection system that harnesses the potential of DCNN for network traffic analysis. The proposed framework is evaluated with four publicly available IDS datasets, namely ISCX-IDS 2012, DDoS (Kaggle), CICIDS2017, and CICIDS2018. Our results demonstrate the effectiveness of the optimized DCNN model in improving IDS performance and accuracy. With detection accuracy levels ranging from 99.79% to 100%, our results underscore the model's efficacy, offering a dependable and efficient approach for the detection of cyber threats. The outcomes of this study have significant implications for network security, providing valuable insights for practitioners and researchers working towards building robust and intelligent intrusion detection systems.

1. Introduction

Network security is a crucial aspect of modern information technology systems, as the increasing reliance on interconnected networks has made them vulnerable to various cyber threats. Unauthorized access, data breaches, and malicious activities pose significant risks to the confidentiality, integrity, and availability of network resources. In response to these threats, IDS have been developed to monitor and analyze network traffic for the detection and prevention of unauthorized or anomalous behavior.

Intrusion detection systems are designed to identify and respond to security breaches in real time by monitoring network traffic and analyzing patterns of activity. They play a crucial role in maintaining the security and integrity of computer networks by detecting various types of attacks, including denial-of-service (DoS), port scanning, and malware infiltration. Traditional IDS approaches rely on predefined rules and signatures to identify known attack patterns, but they often struggle to detect novel or sophisticated attacks.

To address cyber threats challenges, researchers [1–3] have used various datasets and methodologies to improve network security and cloud computing. One such approach is predicting and modeling cyber attacks, which has the potential to enhance cyber security in much the same way that weather forecasting has aided society in reducing the impact of natural disasters.

To overcome the limitations of traditional IDS methods, advanced techniques such as deep learning have emerged. Deep learning algorithms, particularly CNNs, have shown remarkable capabilities in automatically extracting intricate patterns and features from complex data, including network traffic. By leveraging DCNN, intrusion detection systems can effectively identify anomalous behavior and detect emerging threats in real time. These advanced systems offer improved accuracy, scalability, and adaptability, making them a promising solution for enhancing network security and mitigating the risks associated with evolving cyber threats.

Some key important role why Network security and intrusion detection systems play a vital role in safeguarding computer networks and protecting sensitive information are as follows:

* Corresponding author.
E-mail address: vanlalruata.hnamte@gmail.com (V. Hnamte).

https://doi.org/10.1016/j.teler.2023.100077
Received 28 April 2023; Received in revised form 5 June 2023; Accepted 4 July 2023

1. **Prevention of Unauthorized Access**: Network security measures, including intrusion detection systems, help prevent unauthorized access to networks and systems. By monitoring network traffic and identifying potential threats or suspicious activities, these systems can proactively detect and block unauthorized users or malicious intruders.
2. **Early Threat Detection**: Intrusion detection systems are designed to detect and alert administrators about potential security breaches in real time. By continuously monitoring network traffic and analyzing patterns of activity, these systems can identify and respond to threats at their earliest stages, allowing for prompt mitigation actions to be taken.
3. **Protection of Data Integrity**: Data integrity is crucial for maintaining the trustworthiness and reliability of information. Intrusion detection systems can identify attempts to modify or tamper with data, ensuring the integrity of critical assets. By detecting unauthorized modifications or data tampering, these systems help prevent the dissemination of false or malicious information.
4. **Mitigation of Financial Losses**: Cyberattacks can result in significant financial losses for organizations. Intrusion detection systems aid in minimizing these losses by swiftly detecting and responding to security incidents. By identifying and isolating compromised systems or networks, these systems help prevent data breaches, service disruptions, and associated financial damages.
5. **Compliance with Regulations**: Many industries and sectors have specific regulations and compliance requirements regarding network security. Intrusion detection systems assist organizations in meeting these requirements by actively monitoring network activities, detecting potential security breaches, and generating audit logs for compliance reporting purposes.
6. **Proactive Threat Intelligence**: Intrusion detection systems provide valuable insights into emerging threats and attack patterns. By analyzing network traffic and identifying new attack vectors, these systems contribute to the development of proactive threat intelligence, enabling organizations to strengthen their security posture and implement necessary preventive measures.

Implementing robust security measures and utilizing effective intrusion detection systems is essential for safeguarding sensitive information and maintaining the trust of users and stakeholders. For example, the impact of Denial of Service (DoS) attacks and Distributed Denial of Service (DDoS) attacks on the Internet has been extensively documented in the computer networks literature [4–6]. These attacks aim to make a network incapable of delivering regular service by attacking either the network's bandwidth or connectivity. Attackers accomplish their objective by flooding a victim's network or processing capability with packets, denying access to regular users [7].

Deep Convolutional Neural Networks have demonstrated remarkable capabilities in various domains, such as image recognition and natural language processing. The application of DCNN in intrusion detection systems offers several advantages. Firstly, DCNN can learn and adapt to changing attack strategies, providing better resilience against evolving threats. Their ability to process large amounts of data in parallel also enables efficient and real-time analysis of network traffic, ensuring timely detection and response to potential security breaches. Moreover, DCNN-based IDS can reduce false positives by effectively distinguishing between normal network behavior and malicious activities, minimizing the burden on security personnel.

The potential benefits of integrating DCNN into intrusion detection systems are significant. This paper aims to explore the use of DCNN as a novel framework for dependable intrusion detection, providing a comprehensive evaluation of their performance in detecting various types of attacks. Through empirical studies and performance evaluations, we aim to shed light on the effectiveness and practicality of DCNN-based IDS, paving the way for more robust and accurate network security solutions.

This study presents a comprehensive evaluation benchmark for network attacks using publicly available IDS datasets, including ISCX-2012, DDoS (Kaggle), CICIDS2017, and CICIDS2018. The proposed approach utilizes a Deep Learning-based Optimized DCNN for attack detection and compares it with the traditional Deep Neural Network (DNN). The contributions of this paper include:

- Proposing the optimized DCNN model.
- Preparing the training and testing datasets, applying the proposed model for the classification task.
- Evaluating the performance using various datasets.

The structure of this paper is as follows: Section 2 discusses related work in the field. Section 3 presents the methodology used in the proposed model. Section 4 describes the experimental setup for the study. Section 5 discusses the results and comparison of the proposed model with other approaches. Finally, the conclusion section summarizes the study's findings and highlights future research directions.

2. Related works

DL-based IDS can adapt to changing network conditions and detect new threats as they emerge. The need for DL-based IDS has become more pronounced as cyber threats become more sophisticated, and traditional IDS struggle to keep up with the evolving threat landscape. Many studies have investigated the use of DL for detecting DDoS attacks [8]. Table 1 provides the related studies, including the detection and analysis level.

Various studies have explored the use of deep learning (DL) models for detecting DDoS attacks. For example, Grosse et al. [9] used adversarial training to detect malware by first training a deep neural network (DNN) model on the DREBIN dataset, which contains benign and malicious services. They then used generated adversarial scenarios to retrain the model, which improved its generalization and made it less susceptible to adversarial attacks. The analysis for time consumption on the model training and validation was missing. Alzahrani and Hong [10] proposed using artificial neural networks (ANNs) with a signature-based approach to detect DDoS attacks in an intrusion detection system (IDS) on Amazon EMR. Their experiments showed that ANNs outperformed the signature-based method in all scenarios, but, the study did not consider the time consumption on the training and validation.

Zhu et al. [11] suggested analyzing network traffic using DL models such as the convolutional neural network (CNN) and feedforward neural network (FNN) for DDoS intrusion detection. In their experiments on the NSL-KDD dataset, the FNN and CNN models achieved higher accuracy than shallow machine learning techniques like Naive Bayes, Random Forest, J48, Random Tree, and SVM. The result discussions did not include the time consumption analysis for either the training or the validation. Hasan et al. [12] developed a deep convolutional neural network (DCNN) model to detect DDoS attacks on an optical network-like network with a switching scenario. The proposed DCNN achieved 99% accuracy on the BHP dataset, which shallow machine learning techniques failed to analyze effectively. However, there was no detail analysis of time consumption for training and validation. In their study, Li et al. [14] proposed an adversarial-example attack technique using bi-objective GAN to deceive an Android device with an in-cloud firewall. The proposed system achieved 95.2% accuracy, but lack complex analysis on the time utilization.

Krishnan et al. [13] proposed an intrusion detection system that combines a nonsymmetric deep autoencoder (NDAE) DL model with a shallow machine learning method called Random Forest (RF) to ensure SDN security. However, the analysis for model training and validation times was missing. The DL model was chosen to overcome the challenges of shallow machine learning classifiers, such as long training periods, high memory usage, and powerful CPU requirements.

Kushwah and Ranga [15] proposed a novel approach to detect distributed denial-of-service attacks in cloud computing platforms. Their proposed method employs a voting extreme learning machine (V-ELM)

Table 1
Related works on intrusion detection.

No.	Author	Year	Model	Detection Rate	Analysis
1	Grosse *et al.* [9]	2017	DNN	Good	Low
2	Alzahrani and Hong [10]	2018	ANN	Very Good	Low
3	Zhu *et al.* [11]	2018	CNN FFN	Acceptable	Low
4	Hasan *et al.* [12].	2018	DCNN	Good	Low
5	Krishnan *et al.* [13]	2019	NDAE	Very Good	Low
6	Li *et al.* [14]	2020	GAN	Acceptable	Low
7	Kushwah and Ranga [15]	2020	V-ELM	Good	Moderate
8	Velliangiri and Pandey [16]	2020	FT-EHO	Good	Low
9	Novaes *et al.* [17]	2021	GAN	Acceptable	Low
10	Cil *et al.* [18]	2021	DNN	Good	Low
11	Ahuja *et al.* [19]	2021	SVC-RF	Good	Low
12	ElSayed *et al.* [20]	2021	CNN	Very Good	Low
13	Muraleedharan and Janet [21]	2021	DCM	Very Good	Low
14	Hussain and Hnamte [22]	2021	DNN	Good	Low
15	Agrawal *et al.* [23]	2022	M-DBNN	Good	Moderate
16	Priyadarshini and Barik [24]	2022	LSTM	Good	Low
17	Kamel *et al.* [25]	2022	GA	Very Good	Low
18	Yaser *et al.* [26]	2022	DNN-AE	Very Good	Low
19	Alzughaibi *et al.* [27]	2023	M-DBNN	Very Good	Low
20	Thakkar *et al.* [28]	2023	DNN	Good	Moderate
21	Sharma *et al.* [29]	2023	DNN	Very Good	Low
22	El-Ghamry *et al.* [30]	2023	VGG16-PSO	Very Good	Low
23	Wu *et al.* [31]	2023	DNN	Good	Low
24	Hnamte and Hussain [32]	2023	LSTM-AE	Very Good	Good
25	Chanu *et al.* [33]	2023	MLP-GA	Very Good	Good
26	Hnamte and Hussain [34]	2023	DCNNBiLSTM	Very Good	Good
27	This paper	2023	**DCNN**	Very Good	Good

that uses the majority of the outputs from several artificial neural networks (ANNs) to make the final decision. The experimental results suggest that combining multiple ELMs can significantly enhance detection accuracy while reducing false alarms. Velliangiri and Pandey [16] presented a novel approach for DDoS attack detection using a hybrid of fuzzy and Taylor-elephant herd optimization (FT-EHO) inspired by the Deep Belief Network (DBN). The proposed method comprises three modules: feature extraction, feature selection, and classification. The feature extraction module collects features from raw packets, and the feature selection module selects the best features using FT-EHO. Finally, the classification module utilizes FT-EHO to identify DDoS attacks with high accuracy. However, the model training and validation analysis in terms of time consumption was missing.

Novaes et al. [17] proposed an adversarial training-based detection and protection system for SDN that utilizes the GAN architecture to detect DDoS attacks and enhances the proposed system's resilience to conflict attacks. The proposed system has robust modules that enable continuous monitoring of traffic through network flow analysis. However, the model complexity analysis was missing. Cil et al. [18] introduced a DNN-based approach to identify DDoS attacks in an IDS trained on the CICIDS2019 dataset and to monitor malicious network activity. The experiments showed that the proposed model achieved 99.97% accuracy using the first CICDDoS2019 dataset and 94.57% accuracy using the second CICDDoS2019 dataset. However, it is worth noting that using the same dataset on the same model produced different results, which is a clear issue that needs further investigation, also, there was no complexity analysis in the result.

Ahuja et al. [19] proposed a hybrid model of Support Vector Classifier and Random Forest (SVC-RF) for classifying traffic on SDN with high testing accuracy of 98.8% and a low false alarm rate for identifying DDoS attacks that can potentially disrupt the entire network. There was no separate time consumption analysis for the model training and validation. The proposed method utilizes numerous data collected from the switches, which exhibit behavior that is distinct from benign traffic. The OFPFlowStasRequest method in the Ryu application is used to gather different flow-level statistics, while port statistics are obtained by invoking the OFPPortStatsRequest method every 30 seconds and setting them to the training data.

ElSayed et al. [20] proposed a novel hybrid DL method based on convolutional neural networks (CNNs) that achieve up to 97.2 percent accuracy and outperforms single DL models on all assessment criteria. Their approach also allows for a lightweight network intrusion detection system (NIDS) with fewer features without sacrificing performance. Muraleedharan and Janet [21] tackled the challenge of DoS attacks that target the application layer through slow traffic rates and achieved an accuracy rate of 99.63% using a DL model trained on the CICIDS2017 dataset. Though the study was acceptable, the time consumption for training the model and validation was not highlighted.

Hussain and Hnamte [22] proposed a DL-based model to detect attacks in an SDN environment and achieved high accuracy rates of 99.61% and 98.12% on the KDDCup99 and NSL-KDD datasets, respectively. The model was validated using a very old dataset. Agrawal et al. [23] employed a technique called M-DBNN to identify DDoS attacks and made sure that the model's weights were adjusted continuously by using the chimpanzee optimization algorithm. To assess the effectiveness of the weight selection method, they utilized the loss between the model's output and the tag as a measure of fitness. Once the weight selection method was evaluated, the researchers either updated the currently selected weights with a 50% probability or randomly chose another set of weights with a 50% probability. The accuracy of the model increased to 87% after several rounds of experiments. The model training time was 270s whereas the inference time is missing.

In the context of cloud and fog computing security, Priyadarshini and Barik [24] proposed a long short-term memory (LSTM) DL model to detect DDoS attacks at the SDN control layer. The LSTM model was chosen for its ability to handle sequential time-based data, making it suitable for training on collected packets at a specified time interval. However, a details analysis of the time consumption for training and validating the model was not presented. According to Kamel et al. [25], selecting hyperparameters for a model requires significant resources. Genetic algorithms (GA) were proposed and trained with the *DDoS attack SDN* [1] dataset to train the model. Through continuous iterations, individuals within the population gradually approach the optimal solution. The au-

[1] https://data.mendeley.com/datasets/jxpfjc64kr/1

thors utilized GA to modify the data division method and the depth of the decision tree. The study had not considered the time consumption analysis for the model. The model's performance was evaluated using accuracy as a measure of fitness. After several iterations, the accuracy of the model reached 99.46%.

In their paper, Yaser et al. [26] presented an improved method for detecting DDoS attacks using DNNs as Autoencoder. The model was trained on two datasets: ISCX-IDS-2012 and UNSW2018. The model demonstrated high accuracy for the static dataset. Specifically, for the ISCX-IDS-2012 dataset, the model achieved a training accuracy of 99.35% and a validation accuracy of 99.3%. On the other hand, for the UNSW2018 dataset, the model's training and validation accuracy reached 99.95% and 99.94%, respectively. The study had not considered the time consumption for training and validation. Alzughaibi and Khediri [27] proposed DNN Using Backpropagation and trained the model using CSE-CICIDS2018. The model could achieve 98.41% accuracy on multiclass classification. Though the model's performance was acceptable, it was only validated with one dataset, which may not prove the real applicability. Also, the study did not consider the training and validation times which may impact the usability for further research. Thakkar and Lohiya [28] proposed DNN to detect attacks, and trained the model using NSL-KDD, UNSW_NB-15, and CIC-IDS-2017 datasets respectively. The model could achieve 99.84%, 89.03%, and 99.80% respectively in terms of accuracy. Though the detection rate was good, the time consumption for the training is too long, and validation time consumption is not available. Sharma et al. [29] also proposed DNN to detect attacks, where the model was trained with NSL-KDD. The model achieved a 99.3% accuracy rate. The model was not validated with any newer and larger datasets, moreover, the time consumption for training and validation was missing.

El-Ghamry et al. [30] introduced the VGG16-PSO model, which is an optimized CNN model, for intrusion detection. The model was trained and validated using the NSL-KDD dataset, and the results indicate excellent performance. However, it is worth noting that the evaluation was conducted using a dataset that is over two decades old, and only one dataset was used in the study. The study had not mentioned the time consumed for training and validation. In their study, Wu et al. [31] proposed simple DNN to detect network attacks. The model could achieve a 97% accuracy rate. The model was validated only using CSE-CICIDS2018. Though the model was appreciated, the training and inference times are missing. LSTM-AE model was proposed by Hnamte and Hussain [32] to detect attacks, where the model was trained and validated using two publicly datasets; CICIDS2017 and CSE-CICIDS2018. The model could achieve 99.99% and 99.10% accuracy respectively. The model training time was longer and need high-end computing to apply larger datasets.

Chanu et al. [33] proposed MLP-GA to detect attacks. The model could achieve a 98.9% accuracy rate in detection. As the model was not widely validated with newer datasets except CICIDS2017. When the model was validated with newer and larger datasets, the accuracy rate dropped. In their study, Hnamte and Hussain [34] proposed DCNNBiLSTM to detect attacks, trained using CICIDS2018 and EDGE_IIoT[2] datasets. The model could achieve 100% and 99.64% respectively. The study also compared several other commonly used models for detecting attacks, whereas the proposed model achieve the highest in terms of accuracy. The model was also compared between the CPU and GPU in terms of accuracy, loss rate, training time consumption and inference time. Though the model was good at detecting in terms of accuracy, the model was complex and the training time was still too long.

Mauro et al. [35] highlighted that the KDDCUP99 dataset is old and obsoleted. The study reviewed a Neural-based approach for Network Intrusion and briefly explained CNN, RNN, LSTM, and GRU. The study proposed WiSARD for lightweight to handle multivariable datasets. The study compares the classification time with the dataset size which was a good contribution in the field. However, the proposed WiSARD was tested with two datasets only, which almost contain identical data. Dong et al. [36] proposed a semi-supervised Double Deep Q-Network (SSD-DQN) for detecting attacks trained with NSL-KDD and AWID datasets. The model could achieve 79.43% and 98.19% respectively on four classifications with the datasets. As the datasets were older and small in size, the proposed model might not apply in context to modern attacks scenario. In their study, Pelletier et al. [37] used Recurrent Neural Networks (RNNs) and Temporal Convolutional Neural Networks (TempCNNs) to evaluate the training and testing durations. All experiments were conducted on a single machine equipped with 12 Central Processing Units (CPUs) and 256 GB of RAM. Additionally, to assess the computational advantage offered by Graphical Processing Units (GPUs), deep learning experiments were also performed on an NVIDIA Tesla V100 GPU. The outcomes of using GPUs in the experiments highly improved the accuracy rate, as well as shortened the training time comparatively.

The current research has some limitations. For instance, there is no L1 or L2 regularization to optimize the CNN model's hyperparameters when using deeper CNN. However, regularization is not the primary determinant of the model's performance. Instead, the model's architecture plays a more significant role. Previous studies have used CNN and DNN to adjust the model's architecture, but the search space needs to be expanded, and the fitness function must consider the model's running time. To address these issues, this paper proposes a method that overcomes these limitations.

3. Methodology

CNNs use multiple hidden layers to extract features from the data through computations such as convolution, rectified linear units, and fully connected layers [38]. The output layer classifies and identifies the item. CNNs are feedforward networks where information flows from inputs to outputs only in one direction. DNNs are also constructed by combining ANNs without feedback connections. DNNs have input layers, hidden layers, and output layers, with hidden layers potentially having more than three layers. Each layer has units with associated weights, which initiate activation procedures of units from the preceding layer [38–40]. DNNs offer the advantages of both supervised and unsupervised learning. CNNs are feedforward networks that use multiple layers to extract features from data, while DNNs use multiple hidden layers to build complex models capable of supervised and unsupervised learning.

CNN and DNN are both powerful machine learning models but have distinct characteristics and applications. Here are the key differences and uses of CNNs and DNNs:

1. Architecture:
   - **CNNs**: CNNs are specifically designed for processing grid-like data such as images, videos, and audio spectrograms. They consist of convolutional layers that capture local patterns and hierarchical structures in the input data.
   - **DNNs**: DNNs, also known as Multilayer Perceptrons (MLPs), are general-purpose neural networks that can handle arbitrary structured data. They typically comprise multiple fully connected layers that connect every neuron from one layer to the next.
2. Feature Learning:
   - **CNNs**: CNNs excel at automatically learning and extracting meaningful features from raw input data. Through successive convolutional and pooling layers, CNNs can capture hierarchical representations, identifying edges, textures, and higher-level features.
   - **DNNs**: DNNs are effective at learning complex patterns and relationships in data. They can model non-linear relationships between input and output variables, enabling them to capture abstract features and make predictions based on those features.

[2] https://ieee-dataport.org/documents/edge-iiotset-new-comprehensive-realistic-cyber-security-dataset-iot-and-iiot-applications

3. Use Cases:
   - **CNNs**: CNNs are widely used in computer vision tasks, such as image classification, object detection, and image segmentation. They have revolutionized the field by achieving state-of-the-art performance in tasks that involve visual data.
   - **DNNs**: DNNs have a broader range of applications and can be used for tasks such as speech recognition, natural language processing, time series analysis, and recommender systems. They are suitable for tasks that require learning complex patterns and making predictions based on diverse input data.
4. Input Data Size:
   - **CNNs**: CNNs are well-suited for large input data sizes, especially in image processing tasks where the input can have high-dimensional spatial information.
   - **DNNs**: DNNs can handle both small and large input data sizes. They are more flexible in terms of input data shape and can be applied to various domains and problem sizes.

It is worth noting that the specific implementation and details of the DCNN methodology may vary depending on the problem domain, available resources, and the specific goals of the study. However, in this study, we apply L1 and L2 Regularization to optimize the performance of the DCNN.

3.1. L1 Regularization

Lasso, which stands for Least Absolute Shrinkage and Selection Operator, is a type of linear regression that adds a penalty term to the loss function based on the "absolute magnitude" of the coefficients. This penalty term is known as L1 regularization and can be calculated using Eq. 1:

$$L_1 = \lambda \sum_{i=1}^{n} |w_i| \tag{1}$$

where w_i are the coefficients of the regression model and λ is a hyperparameter that controls the strength of the regularization.

3.2. L2 Regularization

Ridge regression is a type of linear regression that adds the "squared magnitude" of the coefficients as a penalty term to the loss function. This penalty term is known as L2 regularization or Tikhonov regularization. The L2 regularization term encourages the model to choose smaller and smoother coefficient values, which can help prevent overfitting. The L2 regularization term is calculated using the following equation:

$$L2 = \lambda \sum_{j=1}^{p} \beta_j^2 \tag{2}$$

where λ is the regularization parameter and β_j represents the jth coefficient of the model.

The key distinction between L1 and L2 regularization is that Lasso (L1) regression tends to shrink the coefficients of less significant features all the way to zero, effectively removing those features from the model. This makes L1 regularization useful for feature selection when there are a large number of features available. Traditional methods for feature selection, such as cross-validation and stepwise regression, perform well when working with a limited number of features, but L1 and L2 regularization is a good choice when dealing with a large number of features.

3.3. Algorithm

The DCNN Algorithm in Algo. 1 is a supervised learning algorithm used for classification tasks, especially in image recognition. The algorithm takes an input image or set of images and classifies them based on their features.

The algorithm starts with the initialization of weight parameters for each layer. The weights are then updated iteratively through a series of epochs. Each epoch consists of a forward pass through the network, where the input image is passed through a series of convolutional, activation, and max-pooling layers to extract its features. The flattened feature map is then passed through a fully connected layer with ReLU activation, and L1 and L2 regularization is applied to control overfitting.

The final layer of the network is a softmax layer, which outputs the probabilities of each class for the given input image. The loss is calculated based on the predicted probabilities and the true labels, and the weight parameters are updated using backpropagation to minimize the loss. This process is repeated for each image in the training set.

After each epoch, the algorithm tests the model's performance on the validation set to monitor its accuracy and ensure that it is not overfitting. At the end of all epochs, the algorithm tests the model's performance on the test set to evaluate its overall accuracy.

By integrating L1 and L2 regularization techniques into the conventional DNN Algorithm presented in Algorithm 2, we sought to enhance its performance. This refined version was subsequently employed for comparative analysis with the proposed DCNN Algorithm outlined in Algorithm 1. The DNN Algorithm encompasses a series of fully connected layers, where each neuron in a given layer is interconnected with all neurons in the following layer. The primary objective of this algorithm is to acquire the most optimal weight parameters and biases for each layer, thereby enabling precise classification of the input data.

Algorithm 1 DCNN Algorithm.

Input: $I = [X_1, X_2, \ldots X_n]$
Output: Classify Attacks W

Initialize weight parameters W for each layer l;
for each epoch $e \in [1, E]$ **do**
  Shuffle the training set
  for each training (x_i, y_i) **do**
    Forward propagation
    Convolution layer: $z^{[l]} = W^{[l]} * a^{[l-1]} + b^{[l]}$
    Activation layer: $a^{[l]} = g(z^{[l]})$ using ReLU
    Max-pooling layer: $a^{[l]} = maxpool(a^{[l-1]})$
    Flatten into a vector $a^{[L]}$
    Fully connected layer with ReLu
    $z^{[L+1]} = W^{[L+1]} a^{[L]} + b^{[L+1]}$
    L1 Regularisation eqn. 1
    L2 Regularisation eqn. 2
    $a^{[L+1]} = softmax(z^{[L+1]})$
    Calculate Loss eqn. 3.
    Update weight parameters
  end for
  Test model performance on validation set
end for
Test model performance on test set

Fig. 1 illustrates the proposed system architecture of the DCNN model, which has an input layer with a unit equivalent to the chosen feature. The input layer uses the ReLU activation function. The CNN layers use 128 and 256 filter units respectively, while the fully connected networks have a denser network with only 256 units and a 0.1 dropout rate that connects to the final layer. Each of the convolutional neural layers uses the ReLU activation function, while the output layer uses the softmax activation function. The categorical cross-entropy loss function is utilized to measure the loss of a sample, which is calculated using Eq. 3. In contrast, Fig. 2 illustrates the architecture of the traditional DNN model used for comparison.

The DCNN model as depicted in Fig. 1 has an input layer that has a unit equivalent to the chosen feature. For the input layer, the relu activation function is used. Layers with 128 and 256 filter units were used in the CNN, while those with fully connected networks had an even

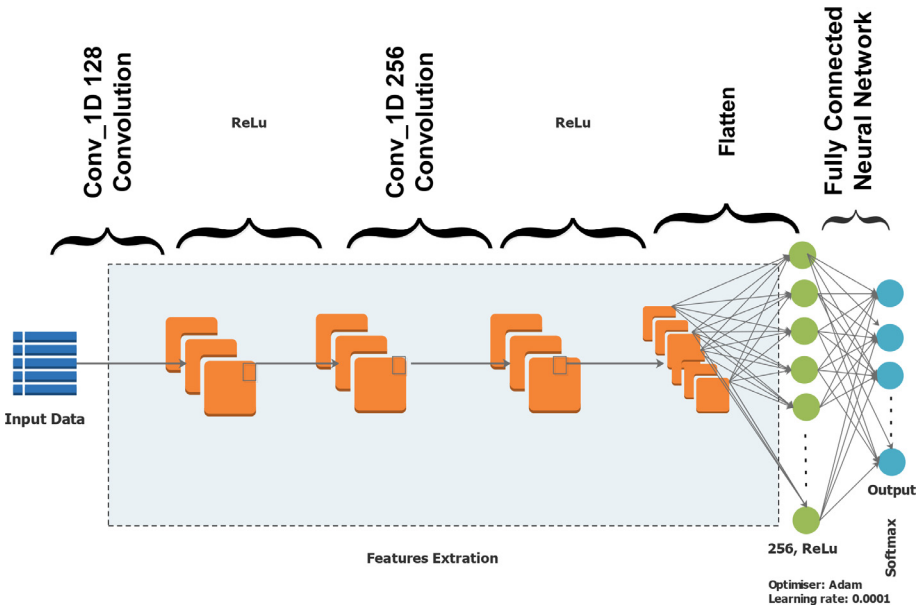


Fig. 1. CNN Architecture.

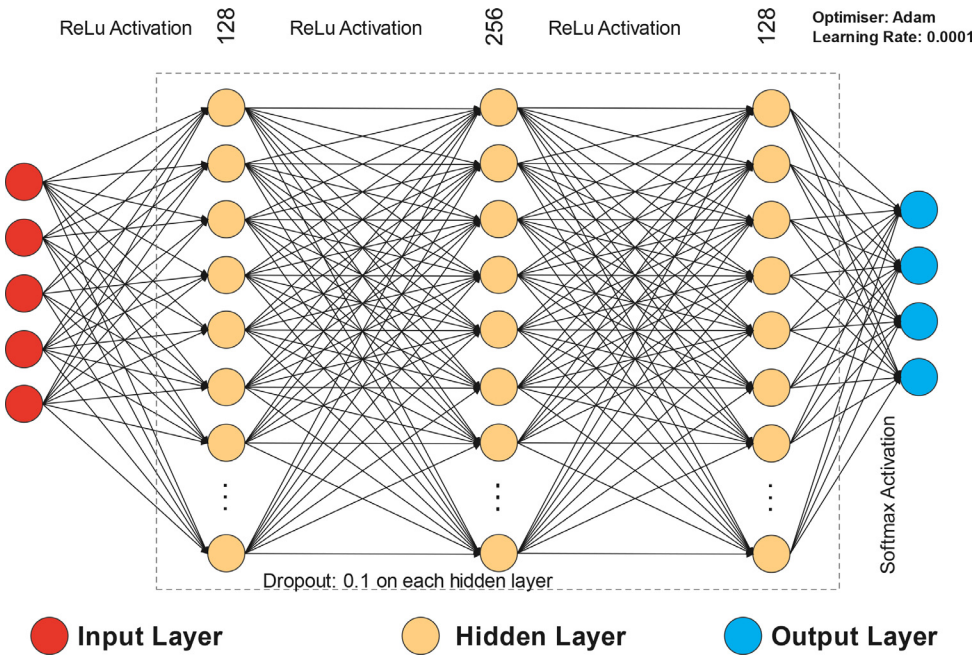


Fig. 2. DNN Architecture.

Algorithm 2 DNN Algorithm.

Input: $I = [X_1, X_2, \ldots X_n]$
Output: Classify Attacks W

Initialize weights and biases $W^{(l)}, b^{(l)}$ for $l \in 1, 2, \ldots, L$ randomly
for epoch in range(epochs) **do**
  for $l \in 1, 2, \ldots, L$ **do**
    Fully connected layer with ReLu
  end for
  L1 Regularisation eqn. 1
  L2 Regularisation eqn. 2
  Calculate Loss eqn. 4.
  Update weight parameters
  Test model performance on validation set
end for
Test model performance on test set

denser network with only 256 units and a 0.1 dropout rate connecting to the final layer. There is a relu activation function utilized in each of the convolutional neural layers and a softmax activation (Eq. 5) in the output layer. The categorical cross-entropy loss function is utilized to measure the loss of a sample by calculating using the Eq. 3 for the DCNN.

$$Loss = -\sum_{i=1}^{output\_size} y_i * log_i \tag{3}$$

i^{th} scalar value in the model output, y_i is the corresponding target value, and $output\_size$ is the number of scaler values in the model output.

$$Loss = -\frac{1}{outputsize} \sum_{i=1}^{outputsize} y_i * \log \hat{y}_i + (1 - y_i) * \log(1 - \hat{y}_i) \tag{4}$$

where $\hat{y}_i$ is the i^{th} scalar value in the model output, y_i is the corresponding target value, and $output\_size$ is the number of scalar values in the

model output.

$$softmax(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}} \quad (5)$$

The DNN model uses the softmax activation function in the output layer, which allows for the calculation of the derivative of the loss function with respect to each weight and each value in the training set using Eq. 4. The relu activation function is used for the other hidden layers, as shown in Eq. 6:

$$ReLu = max(0, x) \quad (6)$$

The relu function returns the input value if it is positive, and 0 otherwise. This helps to introduce non-linearity to the model and improve its ability to learn complex relationships between the input features and the target variable. The optimization algorithm used in this model is Adam, which is an adaptive learning rate method. Unlike traditional gradient descent, Adam calculates individual learning rates for each parameter in the neural network, thereby improving the training process. It adapts the learning rate for each weight by estimating the first and second momentum of the gradient.

The proposed Deep CNN model has larger hidden layers, making it computationally expensive. To improve its performance while conserving computational resources, we can consider reducing the number of layers or using smaller filters in the convolutional layers. Moreover, we can leverage faster hardware, such as GPUs, to speed up the training process. GPUs are specifically designed to accelerate neural network computations and can provide significant speed improvements over CPUs.

To prevent overfitting and optimize the proposed model further, weight regularization can be employed. Weight regularization is a technique that helps reduce overfitting by adding a penalty term to the loss function based on the model's weight complexity. Two common methods of weight regularization are L1 and L2 regularization. L1 regularization adds a penalty proportional to the absolute value of the weights, while L2 regularization adds a penalty proportional to the square of the weights. By incorporating weight regularization into the training process, we can create a more generalizable and accurate model.

4. Experimental setup

In this section, we describe the experimental setup, including the system specifications, dataset characteristics, performance metrics, and experimental outcomes.

4.1. System specifications

The experiments were conducted on a system with the following specifications: Windows 10 Home Edition, Intel Core i9-10900K 4.2 GHz processor, 128 GB RAM, 512GB SSD nvme m.2, and Gigabyte GeForce RTX 3060 Ti (Nvidia) Graphics Card. We used Python 3.9 programming language and deep learning libraries for implementing our proposed model.

4.2. Dataset

To evaluate the effectiveness of our proposed model for intrusion detection, we used some datasets suggested by Hnamte and Hussain [41]. The dataset consists of information collected from various sources, including network traffic flows that contain information about the host, user activity, and system settings.

4.2.1. ISCX 2012

The ISCX IDS 2012[3] dataset provides labeled network traces and complete packet payloads in PCAP format, making it readily available to researchers with the appropriate profiles [42]. The dataset includes profiles for agents that generate real traffic for HTTP, SMTP, SSH, IMAP, POP3, and FTP, which are analyzed to create traces [42]. The dataset contains seven days' worth of network activity, including both normal and malicious activities such as Brute Force SSH, Distributed Denial of Service (DDoS), Botnet, HTTP Denial of Service, and Network Infiltration [42].

4.2.2. DDoS (kaggle)

The Kaggle DDoS dataset[4] consists of publicly available data on DDoS attacks and normal traffic. To add more variation to the dataset, we also incorporated data from other public IDS datasets, including CICIDS2018, CICIDS2017, and CICDoS2016. These datasets include DDoS traffic generated in various years and using different experimental DDoS traffic production technologies. The Kaggle Competition provided two separate datasets: a balanced dataset of 6.32 GB and an imbalanced dataset of 3.83 GB. The dataset contains a variety of features, including Flow ID, Source IP, Source Port, Destination IP, Destination Port, Protocol, Timestamp, Flow Duration, Total Forward Packets, Forward Segment Size Minimum, Active Mean, Active Standard Deviation, Active Maximum, Active Minimum, Idle Mean, Idle Standard Deviation, Idle Maximum, Idle Minimum, and more, for a total of 84 features.

4.2.3. CICIDS2017

The CICIDS2017 dataset[5] is a valuable resource for intrusion detection research as it includes real-world network traffic data in PCAP format. It covers various prominent threats such as DoS, DDoS, brute force, cross-site scripting (XSS), SQL injection, botnet, and port scanning, making it a comprehensive and diverse dataset for evaluating intrusion detection models [43]. The data was collected over five days starting on Monday, July 3, 2017, at 9:00 a.m. and ending on Friday, July 7, 2017, at 5:00 p.m. On Mondays, only regular traffic levels were observed. The dataset contains over 80 network traffic features that were extracted and calculated using the CICFlowMeter open-source software. The dataset is labeled and publicly available on the Canadian Institute for Cyber Security website.

4.2.4. CICIDS2018

The CICIDS2018[6] dataset contains a wealth of information with over 80 network traffic characteristics collected in six columns, including FlowID, SourceIP, DestinationIP, SourcePort, and Protocol [43]. It covers a range of cyber threats such as botnet, Heartbleed, Brute-force, Denial of Service, Distributed Denial of Service, network penetration, and web attacks. The dataset includes 420 computers, including 30 servers and 5 divisions, in the victim's infrastructure. However, the class imbalance in the dataset may pose challenges in using mitigation at the data level, and the data cleansing process for CICIDS2018 is not well documented in recent publications [44]. For our analysis, we chose the ddos-loic-udp_hoic_21-02-2018 dataset for convenience.

4.3. Performance metrics

We evaluated the performance of our proposed system using conventional metrics such as Accuracy (Eq. 10), Precision (Eq. 7), Recall (Eq. 8), and F1 Score (Eq. 9). The Confusion Matrix was used to determine the model's learning requirements. The components of the Confusion Matrix include True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) [45]. The following formulas were used to compute the performance metrics:

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

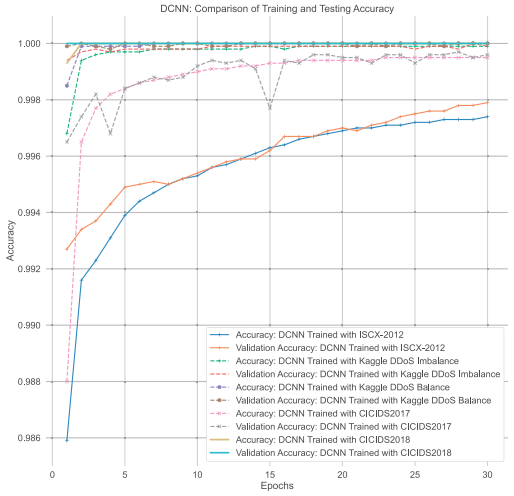
[3] https://www.unb.ca/cic/datasets/ids.html

[4] https://www.kaggle.com/devendra416/ddos-datasets

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

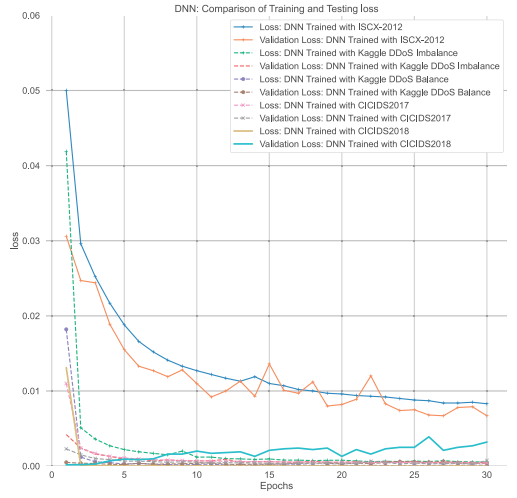
[6] https://www.unb.ca/cic/datasets/ids-2018.html

(a) DNN Accuracy Performance

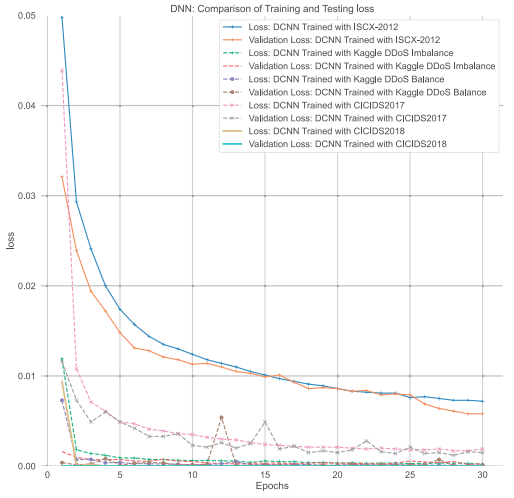


(b) DCNN Accuracy Performance

Fig. 3. Performance Accuracy.



(a) DNN Loss Performance



(b) DCNN Loss Performance

Fig. 4. Performance Loss Rate.

Table 2
Evaluation Result.

No.	Dataset	Size	Model	Accuracy Rate	Loss Rate	Precision	Recall	F-Measure	ROC-AUC
1	ISCX 2012	1.37 GB	DNN **DCNN**	99.76% **99.79%**	0.0067 **0.0058**	99.76% **99.79%**	99.76% **99.79%**	99.76% **99.78%**	97.37% 95.35%
2	DDoS (Imbalance) (Kaggle)	3.83 GB	DNN **DCNN**	99.99% **99.99%**	0.0004 **0.0002**	99.99% **99.99%**	99.99% **99.99%**	99.99% **99.99%**	99.99% 99.99%
3	DDoS (Balance) (Kaggle)	6.32 GB	DNN **DCNN**	100% **100%**	0.0004 **0.0000**	100% **100%**	100% **100%**	100% **100%**	100% 100%
4	CICIDS2017	1.12 GB	DNN **DCNN**	99.80% **99.96%**	0.0007 **0.0015**	99.80% **99.96%**	99.80% **99.96%**	99.80% **99.96%**	95.38% 97.26%
5	CICIDS2018 (LIOS_HOIS)	313 MB	DNN **DCNN**	100% **100%**	0.0032 **0.0000**	100% **100%**	100% **100%**	100% **100%**	100% 100%

$$Recall = \frac{TP}{TP + FN} \tag{8}$$

$$F1 - Score = 2 * \frac{(Precision * Recall)}{Precision + Recall} \tag{9}$$

$$Accuracy = \frac{TN + TP}{TN + TP + FN + FP} \tag{10}$$

$$Y_{t+1} = y_t - \frac{\eta}{\upsilon_t} m_t \tag{11}$$

For optimizing the Adam optimizer, we used the function described in Eq. (11). The learning rate was optimized to 0.0001, which helped us achieve a smaller loss rate. Since υ_t relies on the max operation, it is not advisable to completely bias towards zero as m_t and υ_t in Adam. A lower learning rate made the training more reliable, but optimization was time-consuming due to the small steps required to get the lowest loss function. If the pace of learning is too high, training may not converge or even diverge. The optimizer may exceed the minimum and worsen the loss if weight fluctuations are large [46].

Table 3
Result Comparison with Time Consumption .

Author	Model	Dataset	Accuracy	**Training Time**	**Inferance Time**
Grosse *et al.* [9]	DNN	DREBIN	98.38%	*	*
Alzahrani and Hong [10]	ANN	-	99.98%	*	*
Zhu *et al.* [11]	CNN FFN	NSL-KDD	77.80% 80.34%	*	*
Hasan *et al* [12].	DCNN	BHP	99%	*	*
Priyadarshini and Barik [24]	LSTM	Hogzilla	98.88%	*	*
Krishnan *et al.* [13]	NDAE	CICIDS2017 NSL-KDD	99.24% 99.60%	*	*
Li *et al.* [14]	GAN	E-MalGAN	95.2%	*	*
Kushwah and Ranga [15]	V-ELM	NSL-KDD ISCX	99.18% 92.11%	141.57s 2.76s	*
Velliangiri and Pandey [16]	FT-EHO	KDD-CUP99	93.81%	*	*
Novaes *et al.* [17]	GAN	CICDDoS2019	94.38%	*	*
Cil *et al.* [18]	DNN	CICDDoS2019-1 CICDDoS2019-2	99.97% 94.57%	*	*
Ahuja *et al.* [19]	SVC-RF	Generated	98.8%	*	*
Muraleedharan and Janet [21]	DCM	CICIDS2017	99.61%	*	*
Hussain and Hnamte [22]	DNN	KDDCUP99 NSL-KDD	99.61% 98.12%	*	*
Agrawal *et al.* [23]	M-DBNN	CAIDA DDoS 2007	87%	270s	*
Kamel *et al.* [25]	GA	DDOS attack SDN	99.46%	*	*
Yaser *et al.* [26]	DNN-AE	ISCX-IDS-2012 UNSW2018	99.3% 99.94%	*	*
Alzughaibi *et al.* [27]	M-DBNN	CSE-CICIDS2018	98.41%	*	*
Thakkar *et al.* [28]	DNN	NSL-KDD UNSW_NB-15 CIC-IDS-2017	99.84% 89.03% 99.80%	22318.015s 13913.50s 27719.36s	*
Sharma *et al.* [29]	DNN	NSL-KDD	99.3%	*	*
El-Ghamry *et al.* [30]	VGG16-PSO	NSL-KDD	99.99%	*	*
Wu *et al.* [31]	DNN	CSE-CICIDS2018	97%	*	*
Hnamte and Hussain [32]	LSTM-AE	CICIDS-2017 CSE-CICIDS2018	99.99% 99.10%	184s 462s	53.66s 128.24s
Chanu *et al.* [33]	MLP-GA	CICDDoS 2017	98.8%	45.8s	1.52s
Hnamte and Hussain [34]	DCNNBiLSTM	CICIDS-2018 EDGE_IIoT	100% 99.64%	202s 500s	95.04s 219.29s
This paper	DNN **DCNN**	ISCX2012 Kaggle DDoS - 1	99.76% 99.99% 100%	23s 9s 15s 33s 13s	17.41s 6.65s 11.28s
		Kaggle DDoS - 2 CICIDS2017	99.80% 100% 99.79%	26s 10s 17s 40s 15s	29.05s 9.04s 19.50s
		CICIDS2018 ISCX2012 Kaggle	99.99% 100% 99.96% 100%		7.11s 11.79s 29.36s
		DDoS - 1 Kaggle DDoS - 2			9.91s
		CICIDS2017 CICIDS2018			

Note: * Not available.

5. Results and discussion

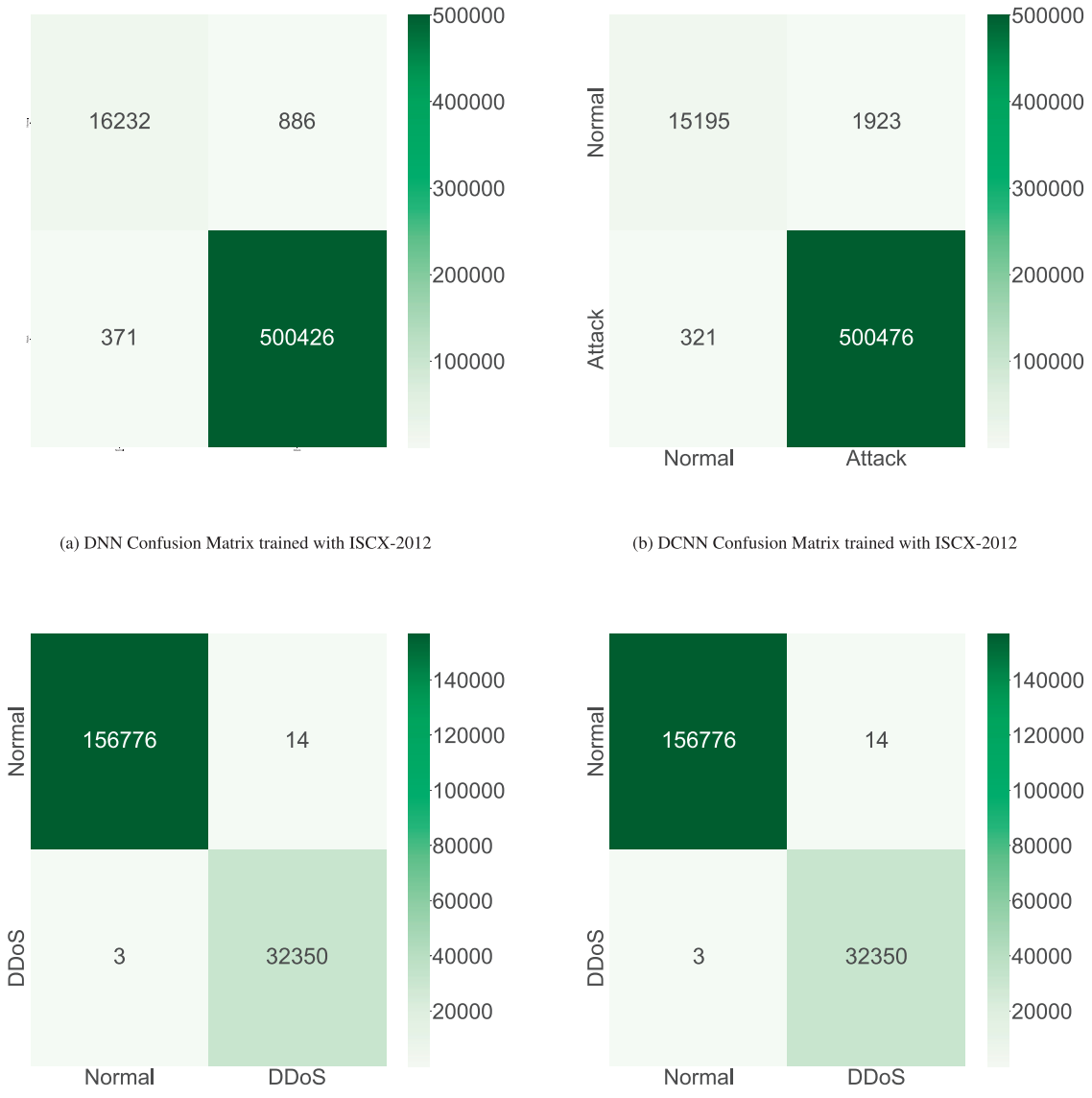
DNN and DCNN require a vast amount of data and features, making hardware acceleration of DNN and CNN computation essential for big datasets. Previous attempts have resulted in the development of hardware platforms that accelerate DNN and CNN computation, achieving high performance and energy efficiency. For example, hardware platforms such as FPGA [47–50] and GPU [51–53] has been used to boost the performance of DNN and CNN. The parallelism of hardware accelerators has enabled significant computational performance gains for DNN and CNN. In this paper, we primarily evaluate the performance of a DNN and CNN accelerator based on a GPU platform and compare its performance across several datasets.

The training of neural networks is a complex and time-consuming process, mainly due to the storage of data and weights in on-chip memory. For large datasets, training a neural network can take days or even weeks, and this necessitates faster and more efficient training techniques, especially for parallel computation of CNNs and DNNs. Hyperparameter optimization is a challenging task during training, as it involves finding the learning rate that ensures convergence to the local minimum without deviating from it. Optimization algorithms adjust the neural network's characteristics, such as weights and learning rate, to minimize losses and produce accurate outcomes. To achieve faster and more efficient training, we used the Adam optimizer with a minimal learning rate of 0.0001 instead of the default setting.

The Processing System (PS) comprises the CPU and off-chip memory and is where data and weights are typically stored in memory, such as Random Access Memory (RAM), from the start. It is difficult to retain all the data in memory due to the large quantity of input data and weights, and hence the CPU may customize the accelerator's control module to support various scales of CONV layers. Basic operations, such as the Softmax function of the CNN model, can also be performed by the CPU. On average, CONV layers account for almost 95 percent of all CNN operations, and CUDA computation significantly speeds up training and reduces total calculation time by using CUDA Unified Memory technology, which merges CPU memory and GPU memory into a single address space.

The field of DL is constantly evolving, and advancements in hardware, particularly the Graphics Processing Unit (GPU), have significantly improved computational efficiency. Given the complexity of DL models, faster and more powerful computing hardware has become essential. GPUs make them ideal for accelerating DL experiments due to the following reasons:

1. **Floating-Point Computing Power**: GPUs are specifically designed to handle intensive mathematical computations, including matrix operations and convolutions, which are fundamental to DL algorithms. GPUs have a large number of cores that can perform computations in parallel, allowing for faster execution of complex DL models. Moreover, GPUs often have dedicated hardware for floating-point operations, enabling efficient and high-speed numerical computations required for DL tasks.
2. **Read/Write Speed**: DL experiments involve processing large amounts of data, such as images, videos, or text. GPUs are equipped with high-speed memory (VRAM) that can efficiently store and retrieve data during computation. This high read/write speed ensures quick access to input data, model parameters, and intermediate results, reducing computational bottlenecks and accelerating DL computations.
3. **Parallelism**: DL algorithms are highly parallelizable, as they involve performing multiple operations simultaneously on different data samples or model parameters. GPUs excel in parallel computing, as they have numerous cores that can execute computations in parallel, enabling concurrent processing of multiple data points or model updates. This parallelism allows for significant speedup in DL training and inference, especially when processing large datasets or complex models.



(a) DNN Confusion Matrix trained with ISCX-2012

(b) DCNN Confusion Matrix trained with ISCX-2012

(c) DNN Confusion Matrix trained with Kaggle DDoS Imbalance

(d) DCNN Confusion Matrix trained with Kaggle DDoS Imbalance

Fig. 5. DNN Confusion Matrix trained with ISCX-2012.

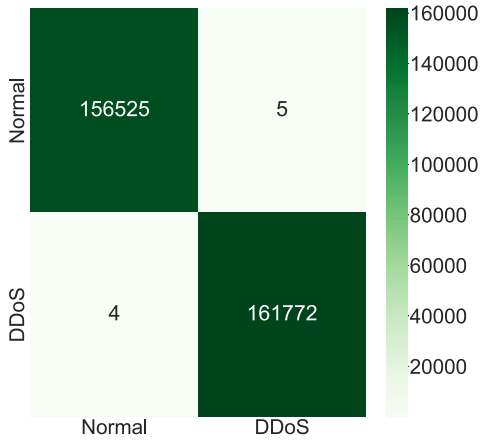
4. **GPU-Specific Libraries and Frameworks**: Several popular DL libraries and frameworks, such as TensorFlow and PyTorch, provide GPU-accelerated implementations of DL algorithms. These libraries leverage the parallel processing capabilities of GPUs to distribute computations across multiple cores, resulting in substantial performance improvements. GPU support in these frameworks simplifies the integration of GPUs into DL workflows and enables seamless utilization of their computational power.

In our experiments, the epoch value for training the model was set to 30 for both the DNN and the CNN. The performance of the three-layer DNN model and the CNN model using selected datasets was almost perfect, as shown in Table 2. The error rates during training for both models were near zero with newer datasets.

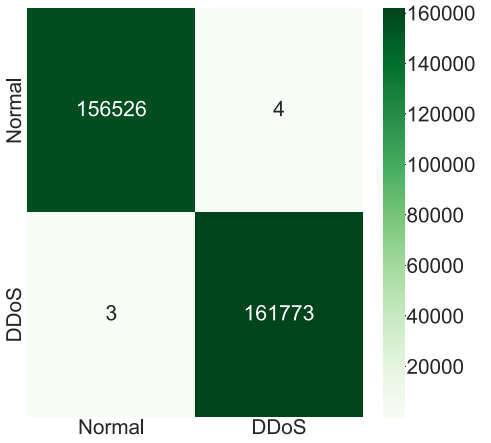
Table 2 presents the evaluation results of the proposed approach for intrusion detection using DCNN. The table consists of several key columns that showcase the performance metrics for different datasets and models. The overall accuracy and loss rate of both models for each dataset used, accurately identifying attacks with a performance ranging from 99.76% to 100%. In addition, the Precision, Recall score, F1-Measure and ROC-AUC of both models are shown. The confusion matrix for the ISCX and Kaggle DDoS (Imbalance and Balance), CICIDS2017, and CICIDS2018 DDoS datasets is shown in Fig. 5.

The accuracy and loss performance of the models is depicted in Figs. 3 and 4, respectively. While the DNN and DCNN models have similar performance on the loss graph, their accuracy performance differs. From Fig. 3, we observe that the DCNN model outperforms the traditional DNN model in terms of accuracy.

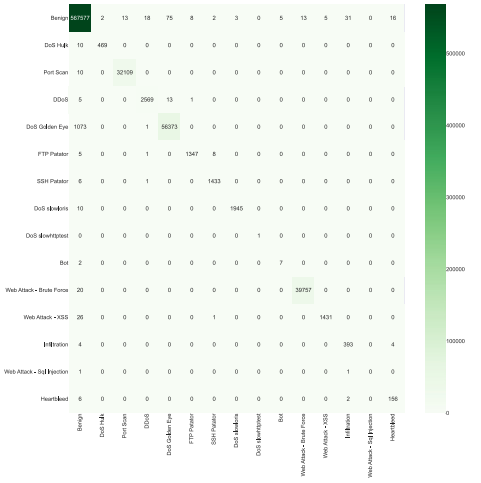
Table 3 provides a comparison of different models and their performance in terms of accuracy and time consumption. Each row represents a specific study or paper that proposes a model for intrusion detection. The table includes accuracy and time consumption results for our proposed model on different datasets, including ISCX2012, Kaggle DDoS - 1, Kaggle DDoS - 2, CICIDS2017, and CICIDS2018 for comparing with the existing study. The accuracy values and corresponding training and inference times are specified for each dataset separately.



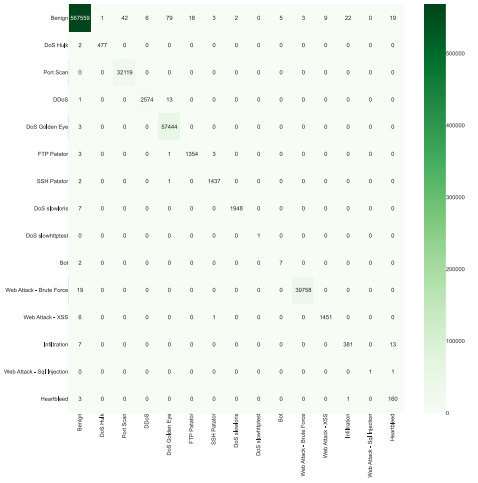
(e) DNN Confusion Matrix trained with Kaggle DDoS Balance



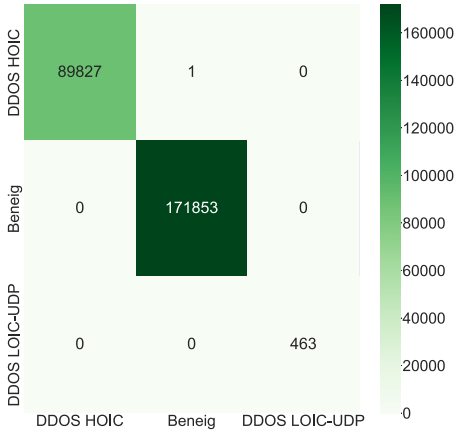
(f) DCNN Confusion Matrix trained with Kaggle DDoS Balance



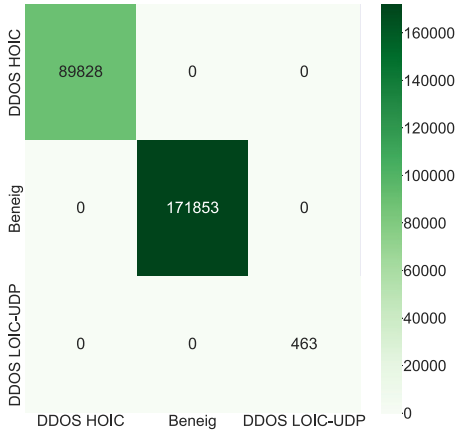
(g) DNN Confusion Matrix trained with CICIDS2017



(h) DCNN Confusion Matrix trained with CICIDS2017



(i) DNN Confusion Matrix trained with CICIDS2018



(j) DCNN Confusion Matrix trained with CICIDS2018

Fig. 5. Continued

To determine how the proposed DCNN is superior to other studies listed in Table 3, we need to analyze the different metrics provided, including the model accuracy, training time, and inference time. Comparing these metrics will allow us to understand the advantages of the proposed DCNN.

1. **Model Accuracy**:
   - The proposed DCNN achieves high accuracy across multiple datasets, ranging from 99.79% to 100%. This indicates that the model is effective in accurately classifying network traffic and detecting intrusions.
   - When compared to other studies, the proposed DCNN consistently achieves high accuracy scores, surpassing many other models in the table.
2. **Training Time**:
   - The proposed DCNN demonstrates relatively fast training times compared to other models. It ranges from 10s to 40s for different datasets.
   - While the table does not provide training times for all models, the proposed DCNN's training time appears to be competitive, as it falls within a reasonable range compared to other models in the table.
3. **Inference Time**:
   - Inference time refers to the time taken by the model to make predictions on new, unseen attacks.
   - The proposed DCNN shows efficient inference times, ranging from 7.11s to 29.36s for different datasets.
   - Similar to training time, the inference time of the proposed DCNN appears to be competitive compared to other models in the table.

Overall, the proposed DCNN stands out due to its high accuracy, competitive training time, and efficient inference time, positioning it as a superior model compared to many others studies mentioned in Table 3.

6. Conclusion

This study presents a novel framework for IDS based on DCNN. The proposed approach offers an effective and reliable solution to detect and classify network intrusions, enhancing the security of computer networks. By leveraging the power of DCNN, the proposed framework achieves improved accuracy and efficiency in detecting network intrusions. The performance evaluation of the proposed framework demonstrates its effectiveness in accurately identifying and classifying various types of intrusions. The evaluation results indicate superior detection capabilities compared to traditional methods, achieving high detection rates and low false positive rates. The framework's ability to handle real-time network traffic and adapt to evolving threats showcases its practicality and reliability in real-world scenarios as shown in Table 3. The model was evaluated using the ISCX2012, DDoS (Kaggle), CICIDS2017, and CICIDS2018 datasets.

Although DCNNs offer promising capabilities for IDS, there are certain limitations and challenges that need to be addressed, such as; the computational resources required for processing large-scale traffic in real-time can be demanding, potentially limiting the practical deployment of DCNN-based IDS in high-speed networks. In the case of an imbalanced dataset, the models loss rate tends to be bigger in comparison to the balanced dataset.

In spite of the above-mentioned limitation, DCNN enables end-to-end learning, meaning they learn directly from raw input data without relying on handcrafted feature engineering. This eliminates the need for manual feature extraction, reducing the laborious and time-consuming process of designing specific features for intrusion detection. The proposed DCNN takes advantage of the parallel computing capabilities of GPUs. Based on our experimental results, the proposed DCNN exhibits several advantages:

- High Accuracy: The proposed DCNN achieves consistently high accuracy across various datasets, outperforming many other models in the table. This suggests that the model is effective in accurately identifying network intrusions.
- Efficient Training Time: The proposed DCNN demonstrates relatively fast training times, indicating that it can be trained efficiently on large datasets.
- Efficient Inference Time: The inference time of the proposed DCNN is also efficient, enabling quick predictions on new, unseen data.

Future research efforts will explore techniques to improve the generalization and transferability of the DCNN model across different network environments and datasets and will focus on developing a DL model for responding to DDoS attacks in real time.

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.

CRediT authorship contribution statement

Vanlalruata Hnamte: Data curation, Writing – original draft, Conceptualization, Methodology, Formal analysis. **Jamal Hussain:** Supervision, Visualization, Writing – review & editing.

Data availability

Data will be made available on request.

Acknowledgment

The authors would like to thank Mizoram University for all the support for this study.

References

[1] H.S. Sánchez, D. Rotondo, T. Escobet, V. Puig, J. Quevedo, Bibliographical review on cyber attacks from a control oriented perspective, Annu. Rev. Control 48 (2019) 103–128, doi:10.1016/j.arcontrol.2019.08.002.
[2] A. Al-Abassi, H. Karimipour, A. Dehghantanha, R.M. Parizi, An ensemble deep learning-based cyber-attack detection in industrial control system, IEEE Access 8 (2020) 83965–83973, doi:10.1109/ACCESS.2020.2992249.
[3] A.J. Gallo, M.S. Turan, F. Boem, T. Parisini, G. Ferrari-Trecate, A distributed cyber-attack detection scheme with application to dc microgrids, IEEE Trans. Automat. Contr. 65 (9) (2020) 3800–3815, doi:10.1109/TAC.2020.2982577.
[4] R.V. Deshmukh, K.K. Devadkar, Understanding ddos attack & its effect in cloud environment, Procedia Comput. Sci. 49 (2015) 202–210, doi:10.1016/j.procs.2015.04.245.
[5] P. Kamboj, M.C. Trivedi, V.K. Yadav, V.K. Singh, Detection techniques of ddos attacks: a survey, in: 2017 4th IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics (UPCON), 2017, pp. 675–679. 10.1109/UPCON.2017.8251130
[6] D. Gautam, V. Tokekar, A novel approach for detecting ddos attack in manet, Mater. Today:. Proc. 29 (2020) 674–677, doi:10.1016/j.matpr.2020.07.332.
[7] A. Bhardwaj, V. Mangat, R. Vig, S. Halder, M. Conti, Distributed denial of service attacks in cloud: state-of-the-art of scientific and commercial solutions, Comput. Sci. Rev. 39 (2021) 100332, doi:10.1016/j.cosrev.2020.100332.
[8] A. Aleesa, B. Zaidan, A. Zaidan, N.M. Sahar, Review of intrusion detection systems based on deep learning techniques: coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions, Neural Comput. Appl. 32 (14) (2020) 9827–9858, doi:10.1007/s00521-019-04557-3.
[9] K. Grosse, N. Papernot, P. Manoharan, M. Backes, P. McDaniel, Adversarial examples for malware detection, in: S.N. Foley, D. Gollmann, E. Snekkenes (Eds.), Computer Security – ESORICS 2017, Springer, Springer International Publishing, Cham, 2017, pp. 62–79. 10.1007/978-3-319-66399-9_4
[10] S. Alzahrani, L. Hong, Detection of distributed denial of service (ddos) attacks using artificial intelligence on cloud, in: 2018 IEEE World Congress on Services (SERVICES), IEEE, 2018, pp. 35–36. 10.1109/SERVICES.2018.00031
[11] M. Zhu, K. Ye, C.Z. Xu, Network anomaly detection and identification based on deep learning methods, in: M. Luo, L.J. Zhang (Eds.), Cloud Computing – CLOUD 2018, Springer, Springer International Publishing, Cham, 2018, pp. 219–234. 10.1007/978-3-319-94295-7_15
[12] M.Z. Hasan, K.Z. Hasan, A. Sattar, Burst header packet flood detection in optical burst switching network using deep learning model, Procedia Comput. Sci. 143 (2018) 970–977, doi:10.1016/j.procs.2018.10.337.

[13] P. Krishnan, S. Duttagupta, K. Achuthan, Varman: multi-plane security framework for software defined networks, Comput. Commun. 148 (2019) 215–239, doi:10.1016/j.comcom.2019.09.014.
[14] H. Li, S. Zhou, W. Yuan, J. Li, H. Leung, Adversarial-example attacks toward android malware detection system, IEEE Syst. J. 14 (1) (2020) 653–656, doi:10.1109/JSYST.2019.2906120.
[15] G.S. Kushwah, V. Ranga, Voting extreme learning machine based distributed denial of service attack detection in cloud computing, J. Inf. Secur. Appl. 53 (2020) 102532, doi:10.1016/j.jisa.2020.102532.
[16] S. Velliangiri, H.M. Pandey, Fuzzy-taylor-elephant herd optimization inspired deep belief network for ddos attack detection and comparison with state-of-the-arts algorithms, Future Generat. Comput. Syst. 110 (2020) 80–90, doi:10.1016/j.future.2020.03.049.
[17] M.P. Novaes, L.F. Carvalho, J. Lloret, M.L. Proença, Adversarial deep learning approach detection and defense against ddos attacks in sdn environments, Future Generat. Comput. Syst. 125 (2021) 156–167, doi:10.1016/j.future.2021.06.047.
[18] A.E. Cil, K. Yildiz, A. Buldu, Detection of ddos attacks with feed forward based deep neural network model, Expert Syst. Appl. 169 (2021) 114520, doi:10.1016/j.eswa.2020.114520.
[19] N. Ahuja, G. Singal, D. Mukhopadhyay, N. Kumar, Automated ddos attack detection in software defined networking, J. Network Comput. Appl. 187 (2021) 103108, doi:10.1016/j.jnca.2021.103108.
[20] M.S. ElSayed, N.-A. Le-Khac, M.A. Albahar, A. Jurcut, A novel hybrid model for intrusion detection systems in sdns based on cnn and a new regularization technique, J. Netw. Comput. Appl. 191 (2021) 103160, doi:10.1016/j.jnca.2021.103160.
[21] M. N, J. B, A deep learning based http slow dos classification approach using flow data, ICT Express 7 (2) (2021) 210–214, doi:10.1016/j.icte.2020.08.005.
[22] J. Hussain, V. Hnamte, Deep learning based intrusion detection system: software defined network, in: 2021 Asian Conference on Innovation in Technology (ASIANCON), IEEE, 2021, pp. 1–6. 10.1109/ASIANCON51346.2021.9544913
[23] A. Agrawal, R. Singh, M. Khari, S. Vimal, S. Lim, Autoencoder for design of mitigation model for ddos attacks via m-dbnn, Wirel. Commun. Mobile Comput. 2022 (2022), doi:10.1155/2022/9855022.
[24] R. Priyadarshini, R.K. Barik, A deep learning based intelligent framework to mitigate ddos attack in fog environment, J. King Saud Univ. - Comput. Inf. Sci. 34 (3) (2022) 825–831, doi:10.1016/j.jksuci.2019.04.010.
[25] H. Kamel, M.Z. Abdullah, Distributed denial of service attacks detection for software defined networks based on evolutionary decision tree model, Bull. Electric. Eng. Inf. 11 (4) (2022) 2322–2330, doi:10.11591/eei.v11i4.3835.
[26] A.L. Yaser, H.M. Mousa, M. Hussein, Improved ddos detection utilizing deep neural networks and feedforward neural networks as autoencoder, Future Internet 14 (8) (2022), doi:10.3390/fi14080240.
[27] S. Alzughaibi, S.E. Khediri, A cloud intrusion detection systems based on dnn using backpropagation and pso on the cse-cic-ids2018 dataset, Appl. Sci. 13 (4) (2023), doi:10.3390/app13042276.
[28] A. Thakkar, R. Lohiya, Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system, Inf. Fusion 90 (2023) 353–363, doi:10.1016/j.inffus.2022.09.026.
[29] B. Sharma, L. Sharma, C. Lal, Anomaly-based dnn model for intrusion detection in iot and model explanation: explainable artificial intelligence, in: S. Rawat, S. Kumar, P. Kumar, J. Anguera (Eds.), Proceedings of Second International Conference on Computational Electronics for Wireless Communications, Springer Nature Singapore, Singapore, 2023, pp. 315–324. 10.1007/978-981-19-6661-3_28
[30] A. El-Ghamry, A. Darwish, A.E. Hassanien, An optimized cnn-based intrusion detection system for reducing risks in smart farming, Internet of Things 22 (2023) 100709, doi:10.1016/j.iot.2023.100709.
[31] C.-s. Wu, S. Chen, A heuristic intrusion detection approach using deep learning model, in: 2023 International Conference on Information Networking (ICOIN), 2023, pp. 438–442. 10.1109/ICOIN56518.2023.10049024
[32] V. Hnamte, H. Nhung-Nguyen, J. Hussain, Y. Hwa-Kim, A novel two-stage deep learning model for network intrusion detection: lstm-ae, IEEE Access 11 (2023) 37131–37148, doi:10.1109/ACCESS.2023.3266979.
[33] U.S. Chanu, K.J. Singh, Y.J. Chanu, A dynamic feature selection technique to detect ddos attack, J. Inf. Secur. Appl. 74 (2023) 103445, doi:10.1016/j.jisa.2023.103445.
[34] V. Hnamte, J. Hussain, DCNNBILSTM: an efficient hybrid deep learning-based intrusion detection system, Telematic. Inf. Rep. 10 (2023) 100053, doi:10.1016/j.teler.2023.100053.
[35] M.D. Mauro, G. Galatro, A. Liotta, Experimental review of neural-based approaches for network intrusion management, IEEE Trans. Netw. Serv. Manage. 17 (4) (2020) 2480–2495, doi:10.1109/TNSM.2020.3024225.
[36] S. Dong, Y. Xia, T. Peng, Network abnormal traffic detection model based on semi-supervised deep reinforcement learning, IEEE Trans. Netw. Serv. Manage. 18 (4) (2021) 4197–4212, doi:10.1109/TNSM.2021.3120804.
[37] C. Pelletier, G.I. Webb, F. Petitjean, Deep learning for the classification of sentinel-2 image time series, in: IGARSS 2019 - 2019 IEEE International Geoscience and Remote Sensing Symposium, 2019, pp. 461–464. doi: 10.1109/IGARSS.2019.8900123.
[38] P. Kim, MATLAB deep learning: with machine learning, in: Neural Networks and Artificial Intelligence, Apress, Berkeley, CA, 2017, pp. 121–147. Ch. Convolutional Neural Network
[39] S.N. Mighan, M. Kahani, A novel scalable intrusion detection system based on deep learning, Int. J. Inf. Secur. 20 (3) (2021) 387–403, doi:10.1007/s10207-020-00508-5.
[40] Z. Ahmad, A.S. Khan, C.W. Shiang, J. Abdullah, F. Ahmad, Network intrusion detection system: a systematic study of machine learning and deep learning approaches, Trans. Emerg. Telecommun. Technol. 32 (1) (2021) e4150, doi:10.1002/ett.4150.
[41] V. Hnamte, J. Hussain, An extensive survey on intrusion detection systems: datasets and challenges for modern scenario, in: 2021 3rd International Conference on Electrical, Control and Instrumentation Engineering (ICECIE), 2021, pp. 1–10. 10.1109/ICECIE52348.2021.9664737
[42] A. Shiravi, H. Shiravi, M. Tavallaee, A.A. Ghorbani, Toward developing a systematic approach to generate benchmark datasets for intrusion detection, Comput. Secur. 31 (3) (2012) 357–374, doi:10.1016/j.cose.2011.12.012.
[43] I. Sharafaldin, A.H. Lashkari, A.A. Ghorbani, Toward generating a new intrusion detection dataset and intrusion traffic characterization, ICISSp 1 (2018) 108–116, doi:10.5220/0006639801080116.
[44] J.L. Leevy, T.M. Khoshgoftaar, A survey and analysis of intrusion detection models based on cse-cic-ids2018 big data, J. Big Data 7 (1) (2020) 1–19, doi:10.1186/s40537-020-00382-x.
[45] D. Powers, Evaluation: from precision, recall and f-measure to ROC, informedness, markedness & correlation, J. Mach. Learn. Technol. 2 (1) (2011) 37–63, doi:10.48550/arXiv.2010.16061.
[46] L.N. Smith, Cyclical learning rates for training neural networks, in: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE, 2017, pp. 464–472. 10.1109/WACV.2017.58
[47] Y. Ma, N. Suda, Y. Cao, S. Vrudhula, J.s. Seo, ALAMO: FPGA Acceleration of deep learning algorithms with a modularized rtl compiler, Integration 62 (2018) 14–23, doi:10.1016/j.vlsi.2017.12.009.
[48] S. Sabogal, A. George, A methodology for evaluating and analyzing FPGA-accelerated, deep-learning applications for onboard space processing, in: 2021 IEEE Space Computing Conference (SCC), 2021, pp. 143–154. 10.1109/SCC49971.2021.00022
[49] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, J. Cong, Optimizing FPGA-based accelerator design for deep convolutional neural networks, in: Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays, 2015, pp. 161–170. 10.1145/2684746.2689060
[50] Y. Ma, Y. Cao, S. Vrudhula, J.s. Seo, Optimizing loop operation and dataflow in fpga acceleration of deep convolutional neural networks, in: Proceedings of the 2017 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, 2017, pp. 45–54. 10.1145/3020078.3021736
[51] S. Dong, P. Zhao, X. Lin, D. Kaeli, Exploring GPU acceleration of deep neural networks using block circulant matrices, Parallel Comput. 100 (2020) 102701, doi:10.1016/j.parco.2020.102701.
[52] J. Li, K.-F. Un, W.-H. Yu, P.-I. Mak, R.P. Martins, An FPGA-based energy-efficient reconfigurable convolutional neural network accelerator for object recognition applications, IEEE Trans. Circuits Syst. II: Express Briefs 68 (9) (2021) 3143–3147, doi:10.1109/TCSII.2021.3095283.
[53] T. Luo, S. Liu, L. Li, Y. Wang, S. Zhang, T. Chen, Z. Xu, O. Temam, Y. Chen, Dadiannao: a neural network supercomputer, IEEE Trans. Comput. 66 (1) (2017) 73–88, doi:10.1109/TC.2016.2574353.