Neural Computing and Applications (2022) 34:15175–15195
https://doi.org/10.1007/s00521-021-06826-6

S.I. : LSNC & OUAI

DFE: efficient IoT network intrusion detection using deep feature extraction

Amir Basati[1] · Mohammad Mehdi Faghih[1]

Received: 24 June 2021 / Accepted: 4 December 2021 / Published online: 29 January 2022

Abstract

In recent years, the Internet of Things (IoT) has received a lot of attention. It has been used in many applications such as the control industry, industrial plants, and medicine. In this regard, a fundamental necessity is to implement security in IoT. To this end, Network intrusion detection systems (NIDSs) have been recently in the detection of network attacks and threats. Currently, these systems use a variety of deep learning (DL) models such as the convolutional neural networks to improve the detection of attacks. However, almost all current DL-based NIDSs are made up of many layers, and therefore, they need a lot of processing resources because of their high number of parameters. On the other hand, due to the lack of processing resources, such inefficient DL models are unusable in IoT devices. This paper presents a very accurate NIDS that is named DFE, and it uses a very lightweight and efficient neural network based on the idea of deep feature extraction. In this model, the input vector of the network is permuted in a 3D space, and its individual values are brought close together. This allows the model to extract highly discriminative features using a small number of layers without the need to use large 2D or 3D convolution filters. As a result, the network can achieve an accurate classification using a significantly small number of required calculations. This makes the DFE ideal for real-time intrusion detection by IoT devices with limited processing capabilities. The efficacy of the DFE has been evaluated using three popular public datasets named UNSW-NB15, CICIDS2017, and KDDCup99, and the results show the superiority of the proposed model over the state-of-the-art algorithms

Keywords Internet of things · IoT · NIDS · Deep learning · Network intrusion detection · Anomaly detection · Convolution neural network · CNN · DFE

1 Introduction

In recent years, the Internet of Things (IoT) has been used in many industries, such as agriculture industries [1], home equipment [2], medical industry [3], and urban development [4]. However, a major issue in IoT networks is to keep the network secure by early detection of cyber-attacks. Network intrusion detection systems (NIDSs) can fulfill this goal by monitoring the network traffic and detecting malicious traffic by applying intelligent algorithms such as machine learning algorithms [5–8]. However, the choice of the proper algorithm is very important as it directly influences the attack detection accuracy and computational complexity of the model. For example, the reference [9] has shown that deep-learning-based NIDS methodologies are preferred nowadays over machine learning methodologies due to their efficiency in learning from large datasets in raw form. On the other hand, there is a trade-off between model computational complexity and the deep structure of deep learning methods. The deeper the algorithm is, the more complex the model will be and hence it will consume more time and computing resources. Another major challenge facing most of the methodologies is their inefficiency in detecting the attack having fewer samples for the training dataset [9]. This class imbalance problem affects the detection rate and accuracy for these minority attack classes, which needs

✉ Mohammad Mehdi Faghih
m.faghih@kgut.ac.ir

Amir Basati
a.basati@student.kgut.ac.ir

[1] Department of Electrical and Computer Engineering, Graduate University of Advanced Technology, Kerman, Iran

further attention. Recently, many deep learning models have been used in NIDSs, including convolutional neural networks (CNNs) [10], auto-encoders [11], deep belief networks (DBNs) [12, 13], recurrent neural networks (RNNs) [14], and long short-term memory networks (LSTMs) [15]. However, in these models, a necessity is that the network should be deep and should have many consecutive layers. Therefore, the memory footprint of these models is very high, and they need a lot of processing resources to achieve good accuracy [16]. Consequently, they are not suitable for use in IoT devices in which memory and processing resources are very scarce [17]. Some methods were recently used to lower the computational complexity of a deep model, including convolution factorization [18] and network compression [19]. A combination of various techniques is also used in our previous work [20] to decrease the computational complexity of the network without losing the classification accuracy. Although current NIDSs have used various methods to reduce their computational complexity and increase their classification performance, they have still many limitations, and there are still room for improvement in multiple aspects, especially in efficiency.

This paper presents a novel and highly efficient network intrusion detection system that can be easily used by IoT devices with limited capabilities. It provides a technique named deep feature extraction (DFE) to extract information from long-range values in an input vector using only a few convolution layers. It also has very high accuracy and a low error rate that makes it suitable for real-time detection of malicious network packets in IoT networks.

In summary, the main proposal and contribution of this paper is to provide a network intrusion detection system for IoT devices that has three main characteristics:

1) It uses deep learning technology.
2) It has a drastically low computational complexity.
3) It provides good attack detection.

To this end, the main idea is to reduce the computational complexity of the network by changing the viewpoint of the network over its input data. Currently, the dominant approach in state-of-the-art methods is to feed the network with 1D or 2D input vectors. Because of this approach, the network should have filters of large sizes or many consecutive layers of small size filters in order to be able to extract discriminant information from distant values in the input vector. On the other hand, the use of a 3D vector as the network input brings the distant values closer together and allows extracting discriminant information using 3D convolution filters. However, the 3D convolution filters make the network more complex and increase its computational needs. Therefore, the main proposal of this paper is to have 2D input vectors for the network, change the 2D input vector to an equivalent 3D one using a clever permutation, and efficiently extract highly discriminant features using only small size 2D convolution filters. In this way, the proposed network is super lightweight and can yet detect the attack types based on the highly discriminant information extracted from all the values in the input vector using a minimal number of learnable parameters.

The rest of the paper is organized as follows: Sect. 2 presents related background information and reviews the existing research. Section 3 represents the proposed method that is consequently evaluated using various measures in Sect. 4. Finally, the paper concludes in Sect. 6.

2 Background

2.1 Deep learning

Deep learning [21] is one of the advanced fields in machine learning that can automatically extract features and model their complex relationships using multiple levels of hierarchical abstractions. It recently becomes prevalent due to its excellence over traditional machine learning algorithms in terms of accuracy. Convolutional Neural Networks (CNNs) are a subset of deep neural network algorithms[22]. Each convolution layer has a set of kernels as learnable weights. Each kernel is convolved with the input of the layer, and the weights in each kernel are reused across the entire input.

In deep neural networks, the problem of vanishing gradient [23] occurs with increasing network depth. It causes convergence issues and reduces the network performance in the CNNs [24]. Residual connection is a technique to overcome the gradient vanishing problem. It is used to allow gradients to flow through a network directly, without passing through non-linear activation functions.

2.2 Related works

Reference [25] presents an IDS named KPCA-DEGSA-HKELM. It uses an extreme learning machine with a hybrid kernel called HKELM. The system uses the Gravitational Search Algorithm combined with the Differential Evolution algorithm to optimize the HKELM parameters. It also introduced the Kernel Principal Component Analysis algorithm for dimensionality reduction and feature extraction of the intrusion detection data. The authors have used the KDDCup99 and UNSW-NB15 datasets to evaluate their work and showed that it has relatively good accuracy in attack detection.

Reference[26] proposed the MSCNN-LSTM integration model that uses a Multiscale Convolutional Neural Network (MSCNN) and Long Short-Term Memory (LSTM)

together. The MSCNN is used to analyze spatial features, and the LSTM is used to extract temporal features. Using these two types of extracted features, intrusion detection is performed, and the famous UNSW-NB15 dataset was used to evaluate the efficiency of the model. The model has been able to reach an accuracy of 89.8% on this dataset.

Reference [27] proposed a method that includes two types of deep and shallow learning. It uses a stacked auto-encoder as a deep learning tool to reduce the dimensions of feature vectors. Then an SVM classifier optimized with the bee colony algorithm is used for attack classification. This paper also used the UNSWNB15 dataset for evaluation, and it has shown that the model can achieve an accuracy of more than 89.62% on this dataset.

In reference [28], a modified Gated Recurrent Unit (GRU) model is used with an SVM classifier as the replacement for the Softmax in the final output layer. Also, the L2-SVM loss function is used instead of the conventional cross-entropy function to improve the efficiency of the model. In order to evaluate the efficiency of the model, a binary classification experiment on the obtained data from the honeypot systems of the Kyoto University was used. The results have shown that the model was able to reach an accuracy of 84.15% on test data and outperforms the conventional GRU-softmax model.

In reference [29], an intrusion detection system is proposed using feature selection and ensemble classifier. In this approach, a dimensional reduction is performed by a heuristic algorithm named CFS-BA, and the optimal feature subset is obtained based on the maximum correlation between features. Then the authors proposed an ensemble approach that combines C4.5, Random Forest, and Forest by Penalizing Attributes algorithms, and at the end, the voting technique is used to combine the probability distributions of the base learners for attack recognition.

a Dew Computing as a Service (DaaS) for improving the performance of intrusion detection in Edge of Things (EoT) ecosystems has been proposed in [30]. It acts as a cloud in the local environment that collaborates with the public cloud to reduce the communication delay and cloud server workload. The paper improved Deep Belief Network (DBN) by a modified Restricted Boltzmann Machine (RBF) and applied it in the real-time classification of attacks. The authors used the UNSW-NB15 dataset to evaluate their work and showed that their proposed method gives good classification accuracy while improving communication latency and reducing the workload of the cloud.

Reference [31] proposed a method named AE-IDS, which is a random forest-based auto-encoder algorithm for network intrusion detection. The technique first constructs the training set with feature selection and feature grouping. After training, an auto-encoder is used for attack detection. In order to evaluate the efficiency of the model, The CSE-CIC-IDS2018 dataset is used, and the results showed that the model is efficient and lightweight as it was able to reduce the amount of computation cost through feature grouping operation as well as the feature selection.

Reference [32] presents the Non-symmetric Deep Auto-Encoder (NDAE) model, which uses shallow learning and a deep neural network in combination. It uses an asymmetric auto-encoder for dimensionality reduction and a random forest classifier for the classification of the network traffic. In order to evaluate the NDAE model, KDDCup99 and NSL-KDD datasets were used, and according to the results, the NDAE model has a lightweight structure suitable for IoT networks.

A multi-stage machine-learning-based NIDS is presented in [33] that tries to reduce computational complexity while maintaining detection accuracy. The paper proposed and compared two feature selection techniques named IGBFS and CBFS and explored their impact on the feature set size, the training sample size, and the detection accuracy. It has shown that the two feature selection methods were able to reduce the feature set size and the training sample size. It has also investigated the impact of machine learning hyper-parameter optimization techniques on the classification performance of K-nearest neighbors (KNN) and the Random Forest (RF) classifiers.

The reference [34] developed an improved deep auto-encoder named Memory-Augmented Auto-Encoder (MemAE) that incorporates a memory module. The idea here is that a normal auto-encoder may generalize so well that it can reconstruct anomalies with small errors, leading to miss-detection of anomalies. In order to mitigate this drawback, this method augments the auto-encoder with a memory module to capture the prototypical elements of the normal data in the training stage. Then at the test stage, the learned memory will be fixed, and the reconstruction is obtained from a few selected memory records of the normal data. The reconstruction will thus tend to be close to normal samples, and the reconstructed errors on anomalies will be strengthened for anomaly detection.

Reference [35] proposed a multi-layer classification approach to identify minority-class intrusions in highly imbalanced data. The method first classifies an input data into normal or attack class. If the data point is an attack, the method then tries to classify it into sub-attack types. The method used a random forest classifier together with a cluster-based under-sampling technique to deal with the class-imbalanced problem.

Reference [20] presents our newly previous work named APAE, based on an Asymmetric Parallel Auto-Encoder. The encoder part of the APAE has a lightweight architecture that contains two encoders in parallel, each one having three successive layers of convolutional filters. The first encoder is for extracting local features using standard

convolutional layers and a positional attention module. The second encoder also extracts the long-range information using dilated convolutional layers and a channel attention module. The decoder part of APAE is different from its encoder and has eight successive transposed convolution layers. The APAE approach has a lightweight and suitable architecture for real-time attack detection and provides outstanding generalization performance even after training using minimal training records. The efficacy of the APAE has been evaluated using three popular public datasets, and the results have shown its superiority over the other algorithms.

Despite existing various NIDSs, they still suffer from two weaknesses: the heavy need for processing resources and poor performance in minority classes. For example, although current deep-learning-based NIDSs provide very good overall classification accuracy, yet they are deep networks that have many successive layers. Therefore, they have many learnable parameters and need a lot of floating-point operations to classify a single input vector. This makes them unsuitable for real-time attack detection in IoT devices with low computing power. In addition, almost all of the existing methods have poor classification accuracy for classes with a small number of records. These minority classes are present in almost all standard datasets, and since they have very few records compared to the entire dataset, existing NIDSs fail to provide accurate classification on them.

3 Proposed approach

This section presents various aspects of the proposed approach. In the following, Sect. 3.1 explains the data preprocessing and the idea behind using 2D data representation. After that, Sect. 3.2 presents the proposed idea of permuting the input vector in 3D space and extracting deep features by regular 2D convolutions.

3.1 Data preprocessing

NIDSs detect intrusion by monitoring data obtained from network traffic. Therefore, they usually capture network traffic and decide to raise the intrusion alarm based on some 1-Dimensional feature vector extracted from network packets. This feature vector usually includes parameters like protocol type, service type, number of failed logins, etc. Current public datasets like KDDCup'99 [36] and CICIDS2017 [37] are also of this form. However, in our previous work [20], we have shown that this 1D representation prevents the convolutional neural networks from achieving high classification accuracy. We have also shown that the order of individual parameters in the input vector of a CNN-based NIDS is important and can highly affect the network computational complexity and classification accuracy. This is because the convolution filters can only extract information from neighboring parameters in the feature vector. In a 1D feature vector, each parameter has only two direct neighbors, while in a 2-Dimensional feature vector, each parameter has eight direct neighbors. Therefore, a 2D filter can extract more neighborhood information than a 1D filter. Thus, the proposed approach uses a 2D representation of vectors as network input. The equivalent 2D form of a 1D input vector with the length L is a square matrix of the size $n \times n$ where $n = \lceil \sqrt{L} \rceil$. Hence, the transformation from the 1D vector to its equivalent 2D matrix can be done by selecting successive groups of n elements from the start of the 1D vector, and placing each group as a row in the 2D matrix. Finally and if needed, the last row of the 2D matrix should be filled with the necessary number of zeros. In addition, some extracted parameters from network packets (like protocol type) are in categorical form, and they should be encoded to integer numbers in order to be possible to use them as network input. To this end, the proposed approach uses "Label Encoding", in which each parameter of categorical type is replaced by its corresponding integer index in an array containing all unique values of that parameter.

3.2 DFE: Deep Feature Extraction

The classification accuracy and computational complexity of a deep learning model highly depend on the structure of that model. The more information the model can extract from its input, the higher the classification accuracy for that model. In contrast, based on the current state-of-the-art approaches, increasing the information extraction capability of the model is possible by deepening the network, which leads to an increase in the computational complexity of the model. Therefore, according to current state-of-the-art approaches, there is a tradeoff between the computational complexity of the network and its capability to extract more information. However, this tradeoff can be overcome to some extent by improving the current state-of-the-art approaches. This fundamental change is the main idea of the proposed approach: changing the viewpoint of the network and forcing it to look at its input data from a different perspective. As explained in our previous work [20], bringing the individual parameters of the input vector by switching from 1D vectors to 2D vectors highly decreases the network computational complexity and improves its classification accuracy. However, in the 2D representation, some parameters are still too far from each other, and the 2D filters cannot extract useful correlation information from them. On the other hand, we can achieve

a much simpler network if we further extend the idea of bringing individual parameters closer together in a 3D space. Consider Fig. 1; if we have 64 features aligned in a 1D vector, an individual parameter has only two direct neighbors. The same individual parameter can have up to eight direct neighbors if we use a 2D alignment and a 3 × 3 filter. On the other hand, if we use a 3*3*3 convolution filter in the 3D space, that individual parameter can have up to 26 direct neighbors. Therefore, the convolution filters can extract correlation information more effectively in the 3D space, and thus, the proposed approach uses the idea of Deep Feature Extraction (DFE): permuting the individual parameters, bringing them closer together in a 3D space, and extracting the correlation information in depth.

Note that it is possible to use 3D convolution filters; however, DFE uses a more straightforward technique based on the standard 2D convolution filters. To explain that, it is necessary to have a brief review of a 2D convolution layer structure. Consider a 2D convolution layer shown in Fig. 2; it has C = 3 channels, each one producing an activation map by applying a convolution filter of the size 3 × 3x2 (the 3rd dimension is equal to the number of channels of the previous layer) on a copy of the layer input. The activation maps together form the output of the layer, which is a 3D structure with the 3rd dimension equal to the number of channels of the layer.

Therefore, the output of a 2D convolution layer is often a 3D structure containing C number of 2D matrices (activation maps), each of which can be seen as a representation of the values in the layer input. As a result, instead of permuting the input vector of the network in a 3D space and applying 3D convolution filters on them, DFE first applies a transfer layer (a 2D convolution layer with the filter size of 1 × 1) on the input vector to obtain a 3D representation. Next, it permutes this 3D representation to obtain a new representation. Then, it applies 2D convolution filters to this new representation. This way, the 2D convolution filters can effectively extract useful correlation information from representatives of the long-range parameters. The proposed model uses two types of permutations to extract the most useful correlation information. The first type of permutation, which is shown in Fig. 3-(a), accepts an input of the size 8 × 8x2 to the size 8 × 2x8. In other words, it takes the corresponding rows of each channel in its input, transposes them, and places them as a channel in its output. The second type of permutation is shown the Fig. 3-(b). It produces the 2 × 8x8 output by taking the corresponding columns of each channel in its input, transposing them, and placing them as a channel in its output.

Using the two mentioned types of permutations, the proposed model brings the long-range features closer together in the channel space. As a result, as shown in Fig. 4, small convolution filters can extract correlation information from long-range parameters in the permuted data.

Figure 5 shows the overall structure of the proposed approach. As you can see, the input to the proposed model is the 2D representation of the input vector obtained by applying the preprocessing procedure described in Sect. 3.1. The proposed approach then transfers the input to four (color-coded) representation channels by applying a transfer layer that has four standard convolution filters of the size 1 × 1. These channels then split into two equal parts to feed the two branches of the model (the blue and yellow channels go to the first branch and the gray and orange channels go to the second branch). Each branch uses one of the permutation types (mentioned in the previous paragraphs) to extract the most useful correlation information. After permutation, a convolution layer with the filters of the size 2 × 2x8 and the stride of 2 is applied on the permutated data in each channel. The result of his convolution in the first branch is of the size 4 × 1x2, and the size of the output of the convolution layer in the second branch is 1 × 4x2. After the convolution layer, another permutation in each branch is applied, and the results of these permutations are concatenated to obtain a new feature

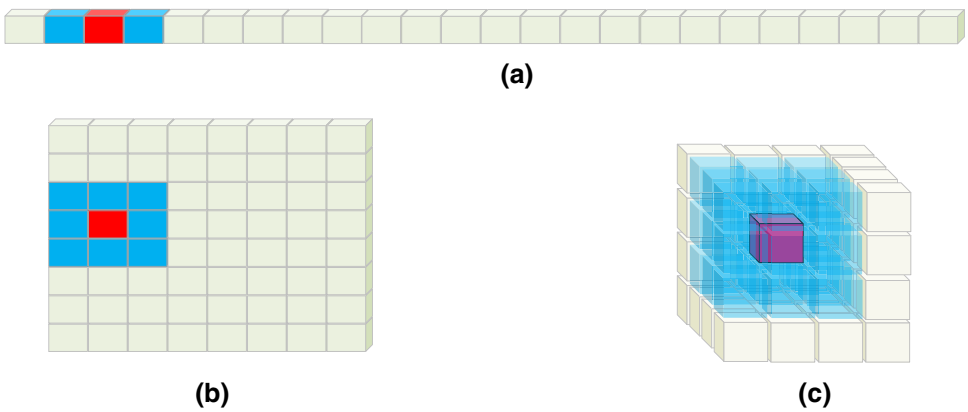


Fig. 1 Direct neighbors in **a** 1D feature vector, **b** 2D feature vector, and **c** 3D feature vector

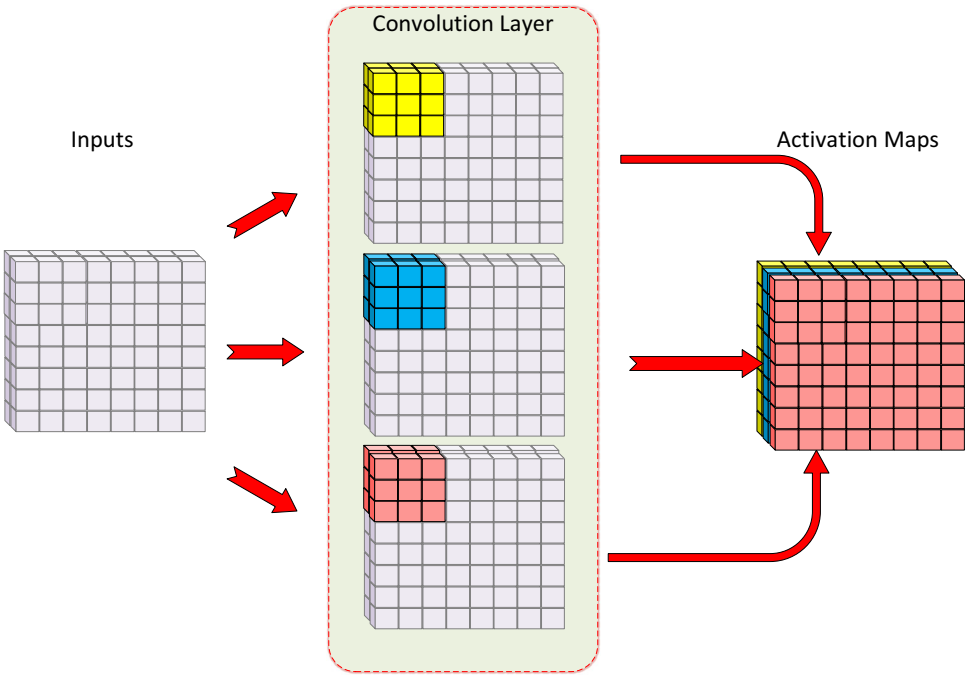


Fig. 2 The general structure of 2D convolution layer

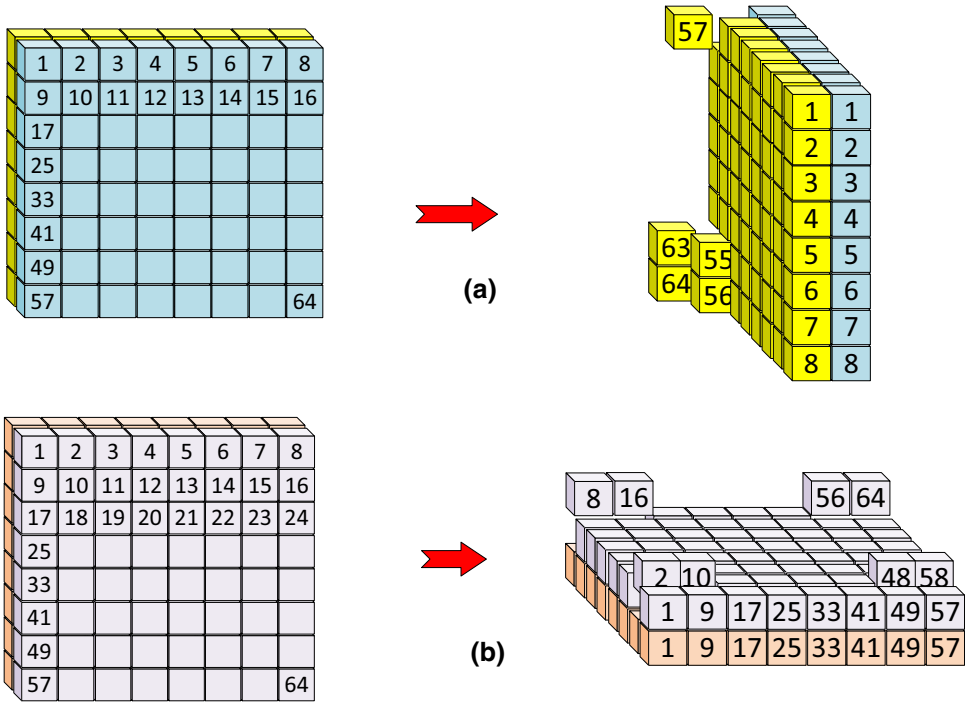


Fig. 3 Two types of permutation in the proposed model. The numbers on the small squares are to show the relative position of the individual parameters

representation. Finally, a booster module is applied to these new features to generate an enhanced feature representation that the classifier uses to perform the attack detection.

The structure of the booster module is shown in Fig. 6. As can be seen, this module augments its input by adding the result of applying four consecutive layers of asymmetric convolutions with the filter sizes of 1×2 and 2×1. The aim of this module is to enhance the features generated with the previous convolution layers and produce a comprehensive feature representation using the least number of learnable network parameters. Because these features are obtained based on the correlation information between nearby and distant values in the input vector, they are very discriminative. They can highly boost the classification accuracy of the classifier.

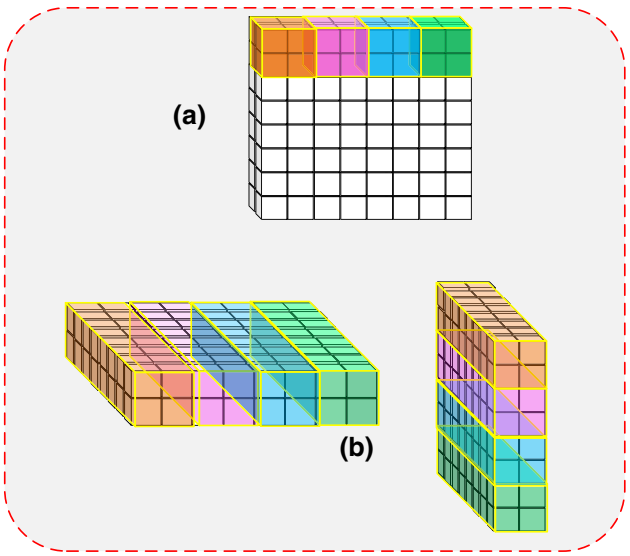


Fig. 4 **a** The application of convolution filters of the size $2 \times 2 \times 2$ on the original data, **b** The application of convolution filters of the size $2 \times 2 \times 8$ on the permuted data. The $2 \times 2 \times 8$ filters can extract correlation information from long-range features in the permuted data. Each colored cube represents a convolution filter

Modern deep learning research has established remarkable achievements in many fields. Yet, the theory behind deep neural networks remains poorly understood. Sure, we understand the math of what individual neurons are doing, but we are lacking a mathematical theory of the developing behavior of the entire network. Lacking a complete theory, we have to rely on intuition. Actually, as stated in the reference [32], researchers are currently unable to explain what makes a successful deep learning structure. In fact, most of the current deep learning models are the result of experiments with numerous structural compositions to achieve the best results. However, the structure of the proposed model has resulted from rational decisions together with experimenting with several structural compositions to achieve the best results. As stated in the first paragraph of this section, the proposed method has a fundamental change in the viewpoint of the network compared to the state-of-the-art methods. This change in viewpoint highly increases the capability of the network to extract discriminant information from its input vector using only a small number of convolution layers. As a result, the proposed approach can provide the same or somewhat better attack detection accuracy than the state-of-the-art methods, while having a fraction of their computational complexity. The proposed approach has significant advantages over the existing NIDSs. First, the proposed approach uses the clever idea of permuting the values of the input vector in the 3D channel space, which allows a single convolution layer with a set of small-size filters to extract strong and discriminative features. This leads to the ability of the classifier to distinguish between various classes of attacks accurately. On the other hand, the permutation operation does not have any learnable parameter, and therefore, it does not add any learnable parameter to the network. The result is a highly compact and efficient network with a small number of parameters that, based on the results of the evaluations in the next section, is highly superior compared to the current state-of-the-art NIDSs. It can accurately discern normal network traffic and different classes of attacks from each other, even in minority classes.

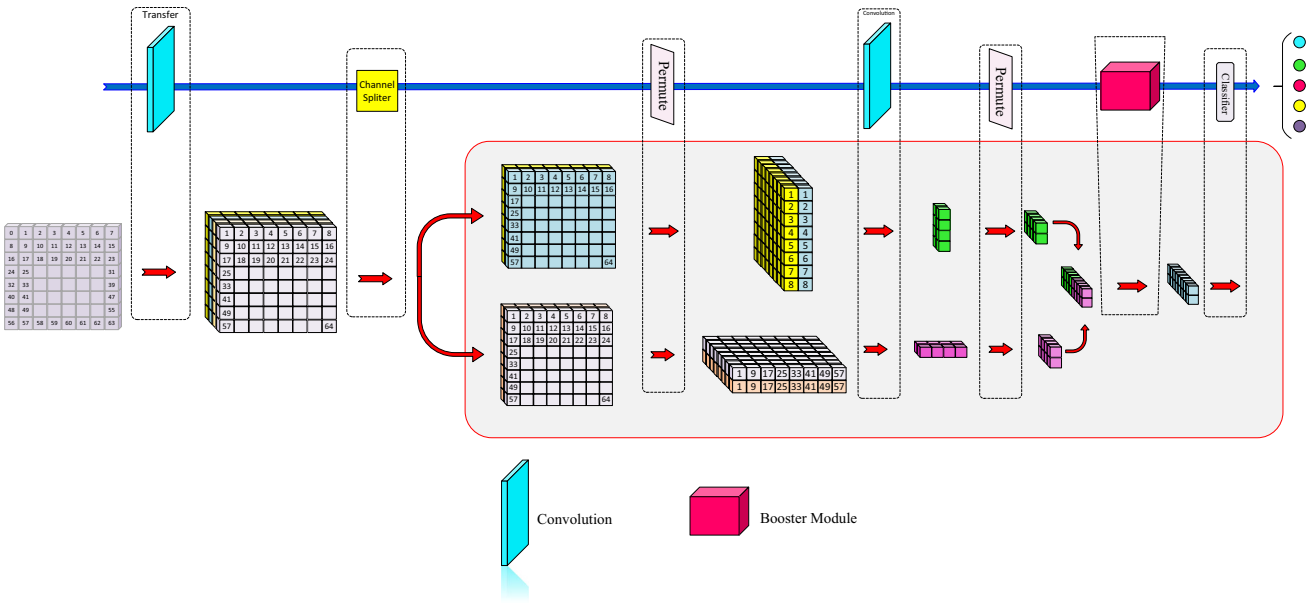


Fig. 5 The overall structure of the proposed model. The numbers on the small squares are to show the relative position of the individual parameters

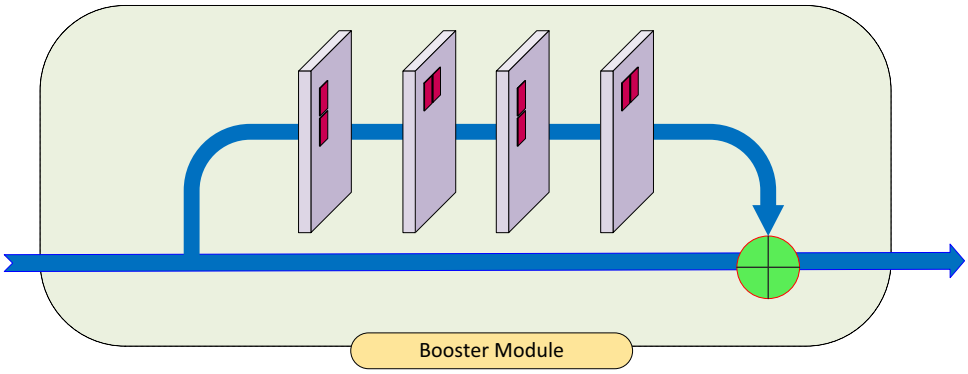


Fig. 6 The structure of the Booster module

4 Results and experiments

The proposed model is implemented with python using the Pytorch library. The Google Colab infrastructure is used for training, and various optimizers have been tested to train the proposed model in 40 epochs. This section presents the results of experiments and compares the proposed approach with the works presented in work [20, 30, 32, 33, 38, 39] and [40] using three public datasets: CICIDS2017, UNSW-NB15, and KDDCup99. Two types of experiments have been done on each dataset. In the first type of experiment (Sects. 4.2 and 4.3), the proposed model has been used as an anomaly detector (binary classification) on each dataset, and in the second type of experiment (Sect. 4.4), the proposed method is used to distinguish between different classes of attack (multiclass classification) in each dataset. Note that the authors of the compared related works did not provide all the needed information for the comparisons with the proposed approach. However, some of them (MemAE [38]) made the source code of their works publicly available (MemAE source code [41]). We also have the source code for our previous approach, which is called the APAE [20]. Therefore, we used these source codes to obtain the required results for the MemAE and APAE algorithms while comparing the proposed approach with other algorithms using only the provided results in their corresponding papers. Note that in the following sections, the "N/A" (Not Available) symbol is used in cases that the source code for a method is not available and the required results for that method have not been provided in the companion paper.

4.1 Datasets

Various datasets exist for the evaluation of network intrusion detection systems. Among existing datasets, the KDDCUP99, CICIDS2017, and UNSW-NB15 datasets are very popular, and many researchers use them for the assessment of their works. Therefore, these datasets are used in this paper for evaluation of the proposed approach, and they are explained in the following three sub-sections.

4.2 KDDCup99

KDDCup99 dataset is a known benchmark dataset in IDS research [40, 42]. This dataset is obtained by processing about 4 gigabytes of compressed tcpdump data collected from 7 weeks of DARPA network traffic. It contains about 5 million feature vectors, and each one represents a single connection record with 41 features, including numeric and categorical features. From these 41 features, three of them are in categorical form and require to be preprocessed with "Label Encoding", as described in Sect. 3.1. After encoding, the feature vectors are padded with zeros and reshaped (as explained in Sect. 3.1 to produce the 2D representation of the size 8×8. The anatomy of this dataset is shown in Fig. 7. As you can see, each vector in this dataset is labeled as Normal or as one of four attack types: Dos, Probe, R2L, U2R. There are also 22 sub-attack types, and each record has labeled with one of them.

Because training a network with this large number of records requires a lot of computational time and resources, it is a common practice to use 10% of the full-size dataset that contains 494,021 training records and 311,029 testing records. However, these sets have a very different distribution of records, i.e. the test set has many records with labels that do not exist in the train set. Therefore, we have split the 494,021 records of the training set into two subsets of 321,113 and 172,908 records, and in the experiments, we have used these new subsets as the train set and the test set, respectively.

4.2.1 CICIDS2017

The CICIDS2017 dataset [37] is one of the most popular databases of IDS research that has been collected at the Canadian Cyber Security Institute. This dataset contains 2,830,473 records that each one has 80 features. 80.30% of the data in this dataset represent benign network

Fig. 7 The anatomy of the KDDCup99 dataset

connections, and the remaining 19.70% of dataset records are network flows that represent six types of common attacks (Dos, Portscan, Infiltration, Web attack, Bot, Brute force) and 14 types of sub-attacks as shown in Fig. 8.

The dataset includes two networks: attack network and victim network. The attack network is a separated infrastructure that has a router, a switch, and a set of attacker PCs with different operating systems executing the attack scenarios. The victim network is a secured network with a firewall, router, switches, and some PCs that each one executes a benign behavior agent. The records of this dataset also need preprocessing, and after applying the mentioned preprocessing in Sect. 3.1, each record is represented by a 2D matrix of the size 9×9. In the experiments of the next sections, a random subset of this dataset with the size of about 65% of the dataset has been used as the train set, and the remaining 35% has been used as the test set.

4.2.2 UNSW-NB15

The UNSW-NB15 dataset [43, 44] has been created in the Cyber Range Lab of the Australian Centre for Cyber Security (ACCS) by generating a hybrid of real normal activities and synthetic contemporary attack behaviors. This dataset has 257,673 records that 175,341 of these records are in the training set, and the remaining 82,332 records are in the testing set. This train and test split configuration is used for the experiments in this paper in order for the results of our work on this dataset to be compatible with the results of previous works. Figure 9 shows the anatomy of this dataset. Among the records in this dataset, 63.9% belong to network flows that represent nine types of attacks (Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, ShellCode, Worms), and the remaining 37.1% of the dataset records represent normal network connections.

Each record of this dataset contains 49 features that two of which are for multiclass and binary labels. From the

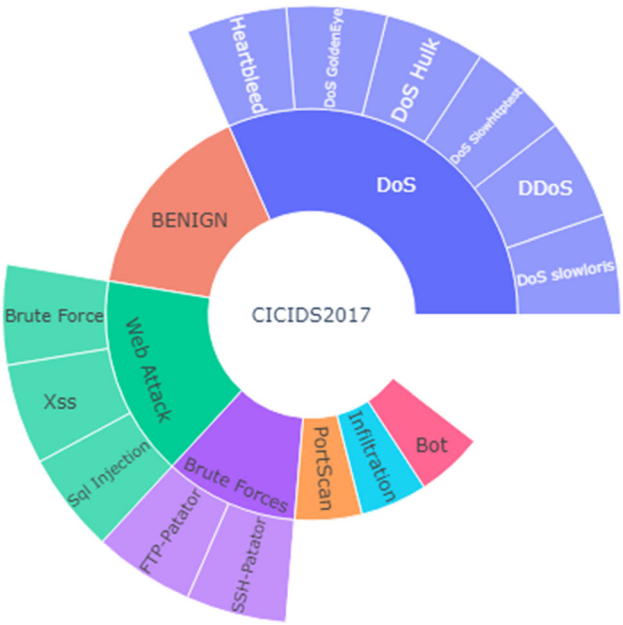


Fig. 8 The anatomy of the CICIDS2017 dataset

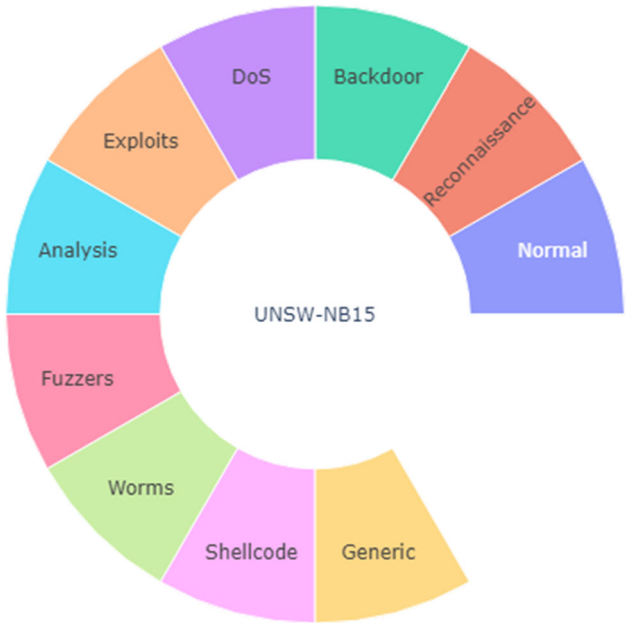


Fig. 9 The anatomy of the UNSW-NB15 dataset

remaining 47 features, 42 features are numeric, and five features are categorical. After applying the preprocessing that has been explained in Sect. 3.1, each record of this dataset can be represented by a 2D matrix of the size 8×8.

4.3 The proposed model trainability

This section is for evaluating the trainability of the proposed model. To this end, three optimizers, each with two different learning rates, have been used to train the proposed model on the three datasets mentioned in Sect. 4.1. Figure 10 shows the results of charts for the training accuracy of the proposed model on three datasets. As you can see, the proposed model can be easily trained to achieve high accuracy with any optimizer within just 40 epochs. However, of the three optimizers, the proposed model can be trained by the ADAM optimizer to achieve an accuracy of near 100% in almost less than five epochs. Therefore, the ADAM optimizer with a learning rate of 0.001 is used for the evaluations in the following sections of the paper.

4.4 Anomaly detection

In this section, the results of the anomaly detection experiment are presented. In this experiment, each record of the three datasets is labeled as either Normal or Attack by combining all attack types into a single attack class for each dataset. Section 4.3.1 shows the results of this experiment on the KDDCUP99 dataset, and the corresponding results for the CDCIDS2017 dataset are also presented in Sect. 4.3.2. Section 4.3.3 also presents the results of this experiment on the UNSW-NB15 dataset.

4.4.1 KDDCup99 dataset

In this section, the anomaly detection performance of the proposed model against other methods is evaluated using the KddCup99 dataset outlined in Sect. 4.3.1. The results of this experiment are presented in Table 1. As you can see, in terms of overall and class-wise classification accuracy, the accuracy of the proposed model is far better than the reference [40]. This is while the accuracy of the proposed method is almost comparable to the APAE and MemAE algorithms. However, the proposed approach significantly

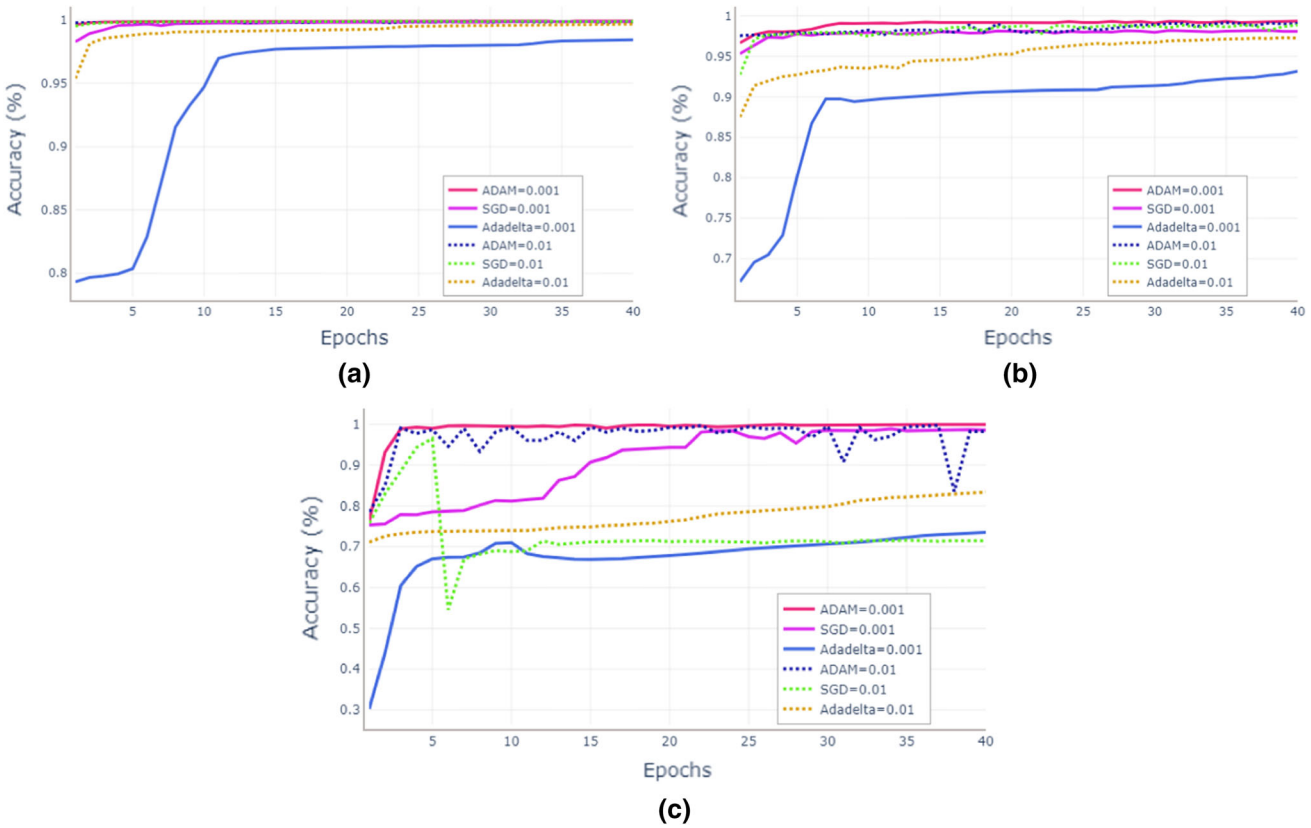


Fig. 10 The accuracy of different optimizers for training the proposed model on different datasets: **a** KDDCUP99 dataset, **b** CICIDS2017 dataset, **c** UNSW-NB15 dataset

Table 1 The results of anomaly detection for KDDCup99 dataset

	Precision (%)			Recall (%)			F-Score (%)		
	DFE	APAE	MemAE	DFE	APAE	MemAE	DFE	APAE	MemAE
Normal	99.76	99.86	99.30	99.81	99.94	99.91	99.78	99.86	99.90
Attack	99.95	99.98	99.98	99.94	99.97	99.83	99.95	99.97	99.97

Overall Accuracy

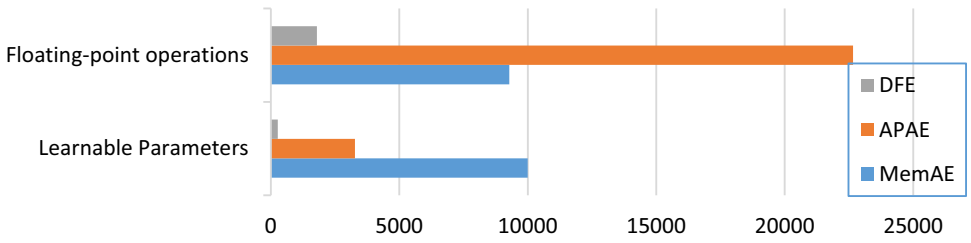
	DFE	APAE	MemAE	[40]	[39]	[30]	[33]	[32]
Accuracy (%)	99.915	99.94	99.84	93.8	N/A	N/A	N/A	N/A

outperforms all algorithms in terms of the number of learnable parameters and floating-point operations. As you can see in Fig. 11, in terms of the number of learnable parameters, the proposed approach has only 266 learnable parameters which shows about 92% and 97% decrease in computational complexity compared to the APAE and MemAE, respectively. This confirms that the proposed approach can be trained with significantly lower training resources than the APAE and MemAE. In addition, the number of floating-point operations of the proposed approach is also considerably lower than other methods. The proposed method can determine the class of an input vector with only 1792 floating-point operations, which are about 92% and 80% lower than the APAE and MemAE, respectively. All of this shows that the proposed approach can provide excellent classification accuracy with a slight computational complexity, and therefore, it is suitable for IoT networks and can be run directly with IoT devices that have minimal computational resources.

4.4.2 CICIDS2017 dataset

Table 2 presents the evaluation results of anomaly detection performance of the proposed approach against other algorithms using the CICIDA2017 dataset outlined in Sect. 4.1.2. The results on this dataset also confirm the superiority of the proposed approach over almost all other methods. The overall classification accuracy of the proposed approach is slightly higher than the APAE and MemAE algorithms. It is a little lower than the accuracy of the presented method in reference [33].

In terms of computational efficiency, the proposed method is again highly superior to the other two methods. Figure 12 compares the performance of the proposed DFE with the other two techniques. As you can see, the number of parameters of the proposed method is about 91% and 98% lower than the number of parameters in APAE and MemAE algorithms, respectively, which shows that the training of the proposed approach needs considerably lower computational resources. The number of floating-point operations for the proposed approach is also significantly lower than the two other methods. The proposed approach only needs 2128 floating-point operations for determining the class of an input vector, which shows about 92% and 84% decrease in computational complexity compared to the APAE and MemAE respectively. This again shows that the proposed method is more suitable for anomaly detection in IoT networks compared to the other two algorithms. It should be noted that in order to obtain the results shown in Table 2, the dataset has been split into two parts. The first part is a random subset of the dataset containing 65% of the total data and has been used for training all three algorithms. The second part also includes the remaining 35% of the data and has been used as the test set for the evaluation of all algorithms.



	Learnable Parameters	Floating-point operations
DFE	266	1,792
APAE	3274	22,656
MemAE	9996	9,279

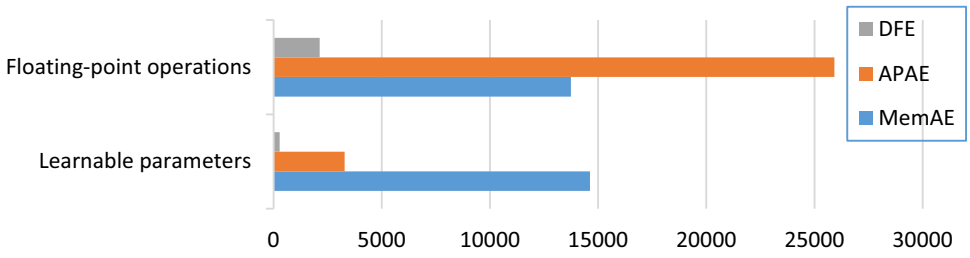
Fig. 11 Performance comparison results for anomaly detection on KDDCup99 dataset

Table 2 The results of Binary Classification for CICIDS2017 dataset

	Precision (%)			Recall (%)			F-Score (%)		
	DFE	APAE	MemAE	DFE	APAE	MemAE	DFE	APAE	MemAE
Normal	99.30	99.38	99.70	98.63	98.04	96.20	98.96	98.71	97.92
Attack	98.69	98.13	96.46	99.32	99.41	99.72	99.00	98.77	98.06

Overall Accuracy

	DFE	APAE	MemAE	[33]	[39]	[40]	[32]	[30]
Accuracy (%)	98.98	99.94	98.19	99.9	N/A	N/A	N/A	N/A



	Learnable parameters	Floating-point operations
DFE	282	2128
APAE	3278	25920
MemAE	14621	13739

Fig. 12 Performance comparison results for anomaly detection on CICIDS2017 dataset

4.4.3 UNSW-NB15 dataset

The anomaly detection results of the proposed approach against the other algorithms using the UNSW-NB15 dataset are shown in Table 3. As you can see, all algorithms (the ones whose corresponding papers have provided the anomaly detection results on this dataset or their source codes are available) have good binary classification accuracy on this dataset. The reason is that the binary classification on this dataset is easy. The reference [33] has calculated the first and second principal components of this dataset and showed that the level of intertwining between the two classes of this dataset is very low, which makes the binary classification of this dataset very easy.

Figure 13 shows the performance comparison for the proposed approach and the other two methods. As it can be seen, the proposed DFE is again highly superior to both APAE and MemAE in terms of efficiency and performance. The proposed model has only 266 learnable parameters that show about 77% and 98% decrease in network computational complexity compared to APAE and MemAE, respectively. The number of floating-point operations of the proposed approach is again considerably lower than the two other methods. It is about 92% and 83% lower than the APAE and MemAE, respectively. This is very important as it again shows that the proposed model can be used in devices with low computational resources, and hence, it better suits IoT networks compared to other NIDSs.

4.5 Multiclass classification

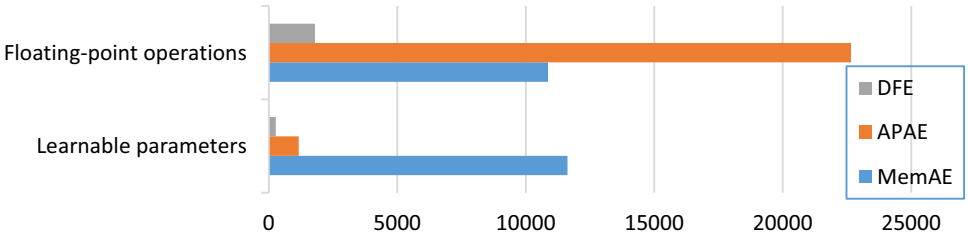
In this section, the results of the multiclass classification experiment are presented. In this experiment, the

Table 3 The results of Binary Classification for UNSW-NB15 dataset

	Precision (%)			Recall (%)			F-Score (%)		
	DFE	APAE	MemAE	DFE	APAE	MemAE	DFE	APAE	MemAE
Normal	100	100	99.99	100	100	100	100	100	99.99
Attack	100	100	100	100	100	99.99	100	100	100

Overall Accuracy

	DFE	APAE	MemAE	[33]	[39]	[30]	[32]	[40]
Accuracy (%)	100	100	99.99	100	N/A	N/A	N/A	N/A



	Learnable parameters	Floating-point operations
DFE	266	1792
APAE	1162	22656
MemAE	11621	10865

Fig. 13 Performance comparison results for anomaly detection on UNSW-NB15 dataset

algorithms are compared based on their capability to determine the true classes of dataset records. Section 4.4.1 shows the results for the KDDCup99 dataset that has five classes: a single class of Normal records, besides four attack classes of U2R, Dos, R2L, and Probe. The results for the CDCIDS2017 dataset are also presented in Sect. 4.4.2, in which there is a total number of seven classes: a single class of Benign records beside six attack classes of Dos, PortScan, Infiltration, Web Attack, Bot, and Brute Force. Section 4.4.3 also shows the results of this experiment on the UNSW-NB15 dataset that has ten classes: a class of Normal records and nine attack classes of Reconnaissance, Backdoor, Dos, Exploit, Analysis, Fuzzers, Worms, Shellcode, and Generic.

4.5.1 KDDCup99 dataset

Table 4 presents the results of the 5-class classification experiment on the KDDCup99 dataset for DFE, APAE, MemAE, and other algorithms. The results show that the DFE is superior to almost all other methods in almost all evaluated class-wise parameters. The proposed method achieved higher values in precision, recall, and F-score for almost all five classes. However, these values for the DFE are almost comparable to the corresponding values for the APAE algorithm. Nevertheless, the notable results are the ones obtained for the minority class: the U2R class. This class has merely 52 records at all, but other classes have many more training records (e.g., the Dos class has 293,582 training records). Therefore, almost all previous NIDSs (except our previous work APAE) have poor classification performance on the U2R class. As you can see in Table 4, the classification performance for the MemAE on the U2R class is very bad. However, the classification performance of the proposed DFE algorithm on the U2R class is very high compared to the MemAE. In addition, the F-Score for the DFE on this class is about 9% higher than the APAE, which shows the advantage of the DFE over the APAE in the classification of classes with uneven distribution. This is also observable from confusion matrixes shown in Fig. 14. The MemAE incorrectly classified all 16 records in the test set of the U2R class, while the classification results of the proposed DFE are correct in 10 cases of a total of 16 test records.

The overall accuracy results on this dataset also confirm the superiority of the proposed approach over the other methods in Sect. 4.5. The DFE accuracy is comparable to the APAE algorithm, while it achieved about 2% superiority over the MemAE method in terms of accuracy. Its

Table 4 The results of multiclass classification for KDDCup99 dataset

	Precision (%)			Recall (%)			F-Score (%)		
	DFE	APAE	MemAE	DFE	APAE	MemAE	DFE	APAE	MemAE
Normal	99.74	99.82	99.66	99.88	99.91	99.51	99.81	99.86	99.58
U2R	90.91	64.71	0	62.50	68.75	0	74.07	66.67	0
DoS	100	100	99.99	100	100	100	100	100	100
R2L	94.85	96.73	70.0	87.16	89.86	89.86	90.85	93.17	78.70
Probe	99.32	98.95	99.90	97.81	98.95	96.48	98.56	98.95	98.16

Overall Accuracy

	DFE	APAE	MemAE	[32]	[40]	[39]	[30]	[33]
Accuracy (%)	99.92	99.94	99.83	98.13	94.2	N/A	N/A	N/A

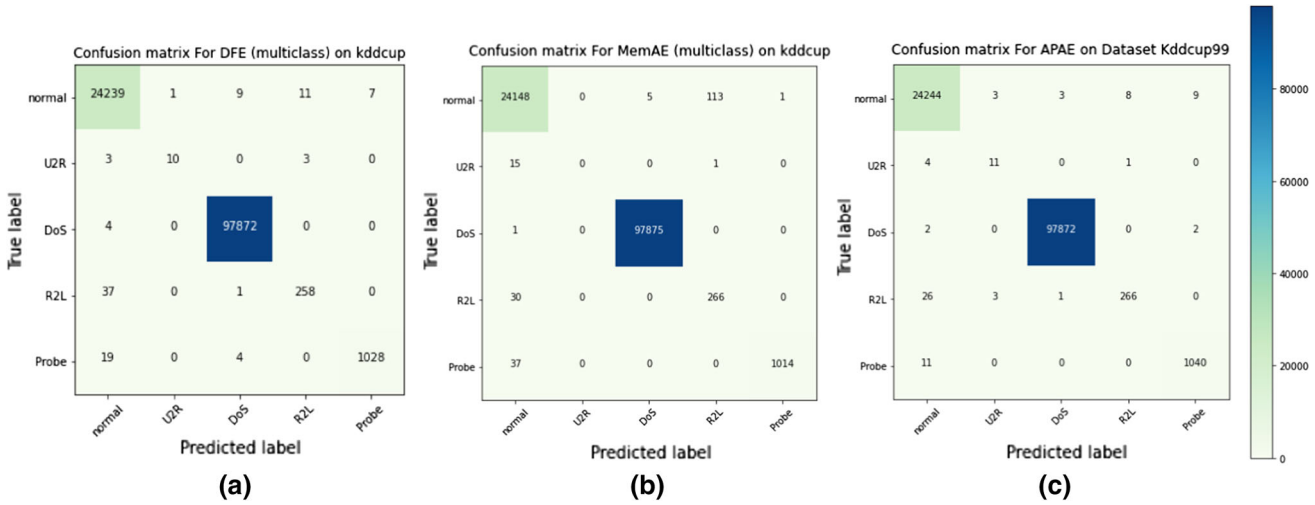


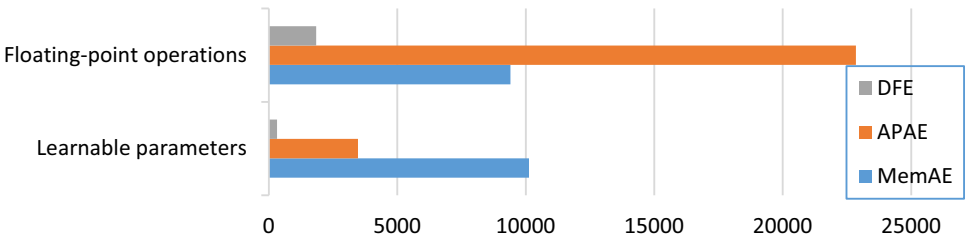
Fig. 14 Multiclass confusion matrix for **a** DFE, **b** MemAE and **c** APAE algorithms on KDDCup99 dataset

accuracy is also far better than the presented methods in reference [32] and [40].

Compared to MemAE and APAE, the DFE again has a significantly smaller number of parameters. As it is shown in Fig. 15, the number of DFE parameters is about 96% and 90% lower than the number of parameters in MemAE and APAE, respectively, which shows the significantly lower needed training resources for DFE compared to the other two methods. Also, the number of floating-point operations of the proposed DFE is again considerably lower than the two other methods. It is about 80% and 92% lower than the number of floating-point operations for the MemAE and APAE, respectively, which shows the advantage of the APAE in terms of performance and its true effectiveness for multiclass attack detection in IoT networks.

4.5.2 CICIDS2017 dataset

Table 5 shows the results of the 7-class attack detection experiment on the CICIDS2017 dataset for DFE against other algorithms. Again, the results show that the DFE is comparable to the APAE while it has dominance over the other methods in terms of overall accuracy. The DFE has achieved an overall accuracy of 99.31%, which is about 1% and 2% higher than MemAE and the reference [39], respectively. This is while the DFE is highly efficient compared to the APAE and MemAE algorithms, and the number of parameters in the DFE is significantly lower than the number of parameters in the other two. This is shown in Fig. 16. As you can see, the DFE has only 367 parameters, which shows a 90% improvement in training efficiency compared to the APAE that has 3599 parameters. The DFE is also more train efficient than the MemAE, and it has about 97% lower number of learnable parameters than the MemAE, which has 15,016 parameters. The number of floating-point operations of the DFE is again much lower than the two other methods. It is about 91% and 84% lower than the number of floating-point operations for the APAE and MemAE, respectively. This again verifies the superior performance of the APAE and its true



	Learnable parameters	Floating-point operations
DFE	317	1840
APAE	3469	22848
MemAE	10122	9402

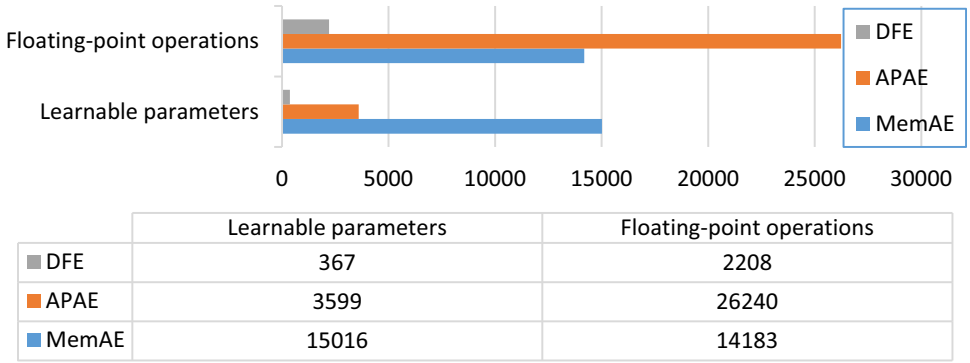
Fig. 15 Performance comparison results for multiclass classification on KDDCup99 dataset

Table 5 The results of multiclass classification for CICIDS2017 dataset

	Precision (%)			Recall (%)			F-Score (%)		
	DFE	APAE	MemAE	DFE	APAE	MemAE	DFE	APAE	MemAE
Benign	99.47	99.81	99.39	99.17	99.21	97.23	99.32	99.51	98.30
DoS	99.06	99.47	98.90	99.76	99.84	99.77	99.41	99.65	99.34
PortScan	99.70	99.58	93.99	99.81	99.88	99.83	99.75	99.73	96.82
Infiltration	100	38.89	0	41.67	58.33	0	58.82	46.67	0
Web Attack	95.63	64.37	88.24	83.06	96.84	9.48	88.90	77.33	17.12
Bot	77.75	82	83.89	51.80	85.93	60.03	62.17	83.92	69.98
Brute Force	98.16	99.38	91.85	95.95	99.75	99.40	97.04	99.57	95.48

Overall Accuracy

	DFE	APAE	MemAE	[39]	[33]	[30]	[40]	[32]
Accuracy (%)	99.31	99.50	98.26	97.90	N/A	N/A	N/A	N/A



	Learnable parameters	Floating-point operations
DFE	367	2208
APAE	3599	26240
MemAE	15016	14183

Fig. 16 Performance comparison results for multiclass classification on CICIDS2017 dataset

effectiveness for multiclass attack detection in IoT networks.

Almost all class-wise parameters for DFE are also comparable or higher than the other algorithms. However, the results for Infiltration and Web Attack (which are two minority classes) classes are considerable. Note that the data distribution between various classes of this dataset is also different. For example, the Infiltration class has only 36 records at all, while the Portscan class has 158,930 records. Therefore, almost all previous NIDSs, except APAE, have poor classification accuracy on the Infiltration class. As you can see in Table 5, the classification performance of the proposed APAE algorithm on the Infiltration class is very high compared to the MemAE. The shown confusion matrixes in Fig. 17 also confirm that the MemAE incorrectly classified all 12 records in the test set of Infiltration class while the classification results of the DFE are correct in 5 cases of a total of 12 test records. The case for the Web Attack class is also similar to the Infiltration class. In the Web Attack class, the MemAE has better precision compared to the APAE. In addition, the F-Score for the proposed DFE is about 9% and 11% higher than the F-Score of the APAE and MemAE, respectively, which shows the advantage of the DFE over the other two algorithms in the classification of classes with uneven distribution.

4.5.3 UNSW-NB15 dataset

The classification results for the 10-class attack detection experiment on the UNSW-NB15 dataset are shown in Table 6. As you can see, the DFE defeats the APAE and MemAE in almost all criteria for individual classes. Note that the distribution of records in this dataset is also unbalanced. For example, the Worms and Shellcode classes are minority compared to other classes. The Exploits class has 44,525 records, while the number of records in the Worms class is only 130 records in the train set and 44 records in the test set. The Shellcode class also contains only 1133 records in the train set and 378 records in the test set. Therefore, the correct classification for these classes is a challenging task, and as you can see from Table 6, the MemAE has poor classification results on these classes. This is while the DFE classification performance on these classes is very significant. It perfectly classified all the records in these two classes. However, in the Worms class, the MemAE has very false positives, and therefore, its precision is very low. In the case of the Dos class, the

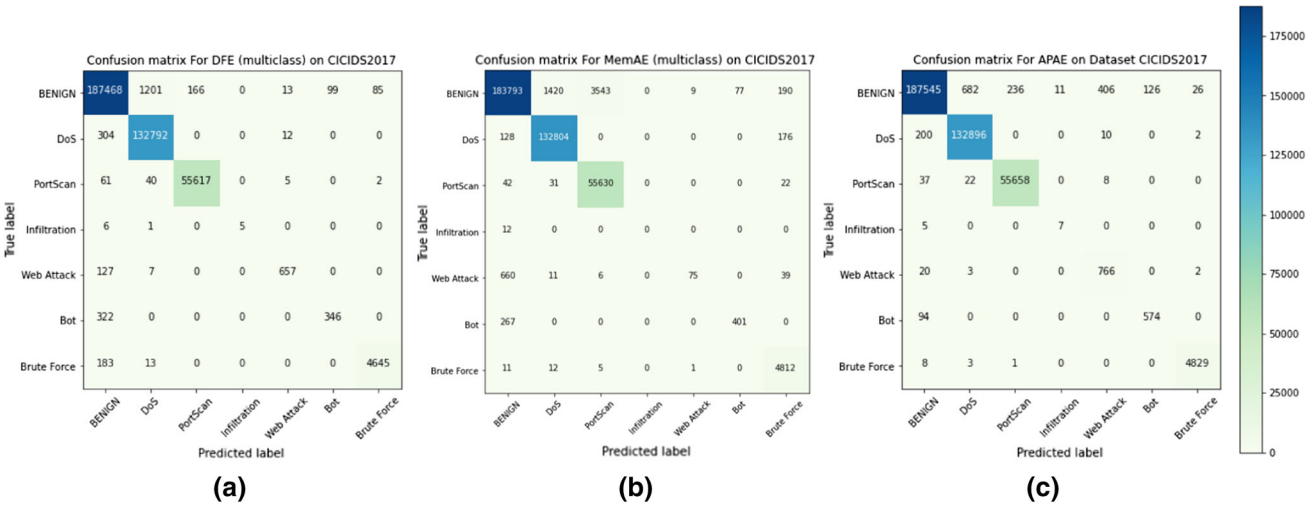


Fig. 17 Multiclass confusion matrix for (**a**) DFE, **b** MemAE and **c** APAE algorithms on CICIDS2017 dataset

Table 6 The results of multiclass classification for UNSW-NB15 dataset

	Precision (%)			Recall (%)			F-Score (%)		
	DFE	APAE	MemAE	DFE	APAE	MemAE	DFE	APAE	MemAE
Normal	100	99.96	99.81	100	100	100	100	99.98	99.91
Reconnaissance	100	100	100	99.97	99.80	96.28	100	99.90	98.11
Backdoor	100	98.31	88.08	100	100	91.25	100	99.15	89.64
Dos	99.42	99.46	85.84	99.98	99.56	98.26	99.70	99.51	91.63
Exploit	100	100	98.91	99.78	99.53	93.70	99.89	99.77	96.24
Analysis	99.85	96.98	90.35	100	99.56	99.56	99.93	98.25	94.73
Fuzzers	100	99.69	99.77	99.98	99.84	98.14	100	99.76	98.94
Worms	100	91.67	8.19	100	100	84.09	100	95.65	14.92
Shellcode	99.74	100	52.78	100	100	5.03	99.87	100	9.18
Generic	100	100	99.89	100	100	99.88	100	100	99.89

Overall Accuracy

	DFE	APAE	MemAE	[30]	[39]	[40]	[33]	[32]
Accuracy (%)	99.96	99.89	98.23	85.71	93.40	N/A	N/A	N/A

MemAE has a high number of false negatives, and as a result, it has a very low recall. The confusion matrixes of Fig. 18 also confirm that the MemAE has a good recall on the Worms class: it has correctly classified 37 records from 44 test records of this class. However, MemAE has very bad precision on this class because of its high False Positive on this class: it has incorrectly classified 415 (sum of the Worms column in Fig. 18 minus 37) records of other classes into the Worms class. The MemAE results for the Shellcode class are also bad: it has correctly classified only 19 records of all 378 records in the test set of this class, which makes it have very bad precision and recall on this class. The DFE also outperforms all other algorithms in terms of the overall classification accuracy. Its overall accuracy is about 1% and 2% higher than the APAE and MemAE, respectively. It is also far better than the overall accuracy of the references [30] and [39].

Note that the classification results of the DFE are very similar to the results of our previous work APAE. However, the DFE is significantly more efficient than the APAE in terms of the number of learnable parameters and the number of floating-point operations. Figure 19 shows the comparison between these values for the three algorithms. As you can see, the DFE has only 402 parameters, which shows about 89% and 96% improvement in training efficiency compared to the APAE and MemAE, respectively. The number of floating-point operations of the DFE is also much lower than the two other methods. The DFE has only 1920 floating-point operations, which shows about 91% improvement in runtime efficiency compared to the APAE

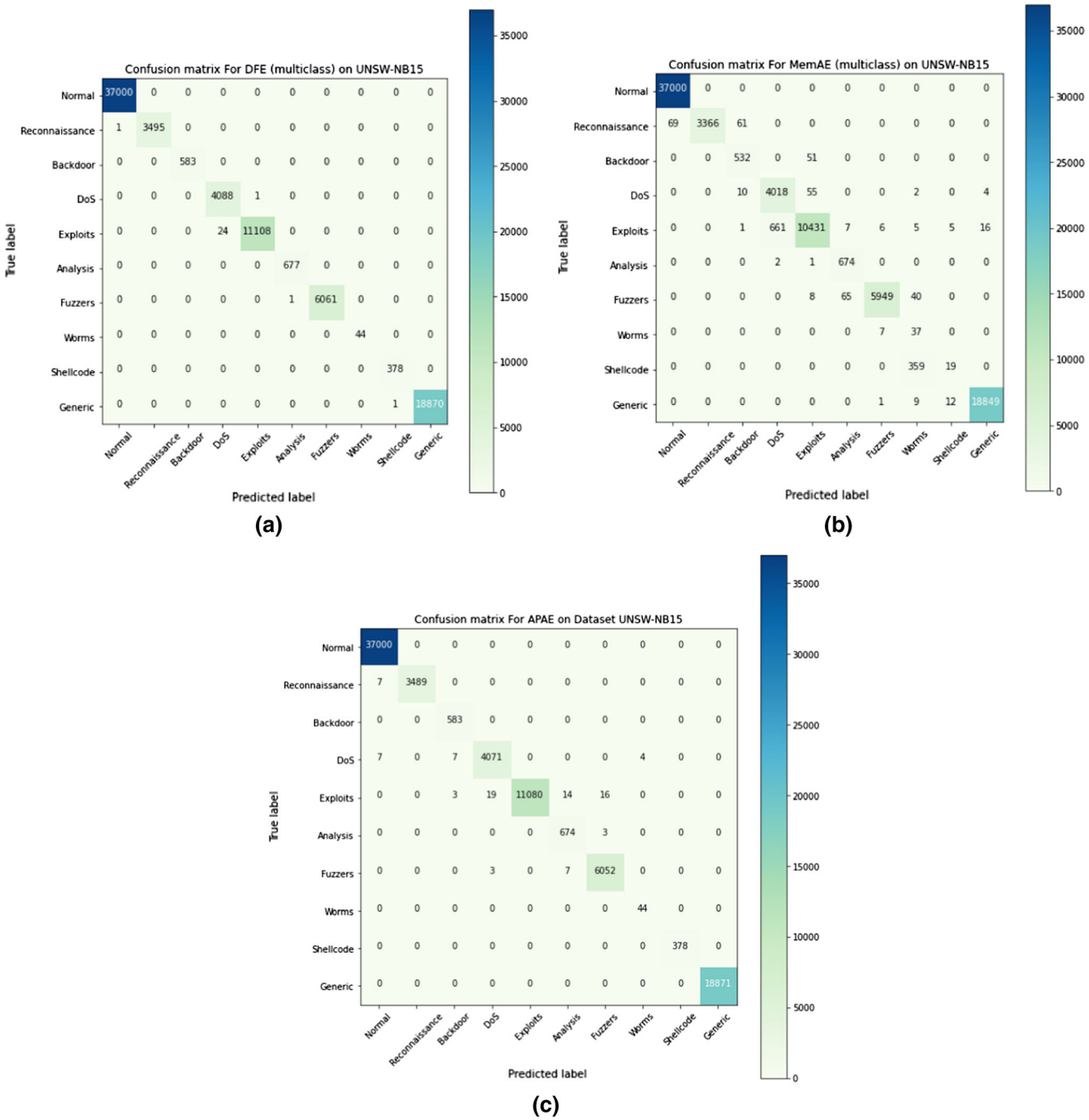
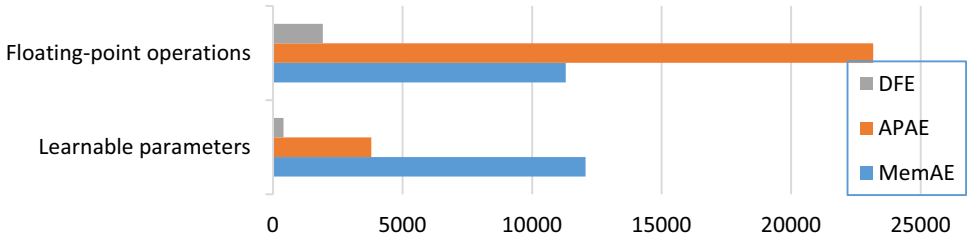


Fig. 18 Multiclass confusion matrix for **a** DFE, **b** MemAE and **c** APAE algorithms on UNSW-NB15 dataset

that has 23,168 floating-point operations. The DFE is also more efficient than the MemAE, and it has about 83% lower number of floating-point operations than the MemAE, which has 11,297 floating-point operations. Again, this comparison verifies the superiority of the DFE and its true effectiveness for multiclass attack detection in IoT networks.

4.6 Empirical performance evaluation

Theoretically, the computational cost of a method depends on the number of needed calculations by that method. Also, the number of required calculations for a method can be represented by its number of Floating-Point Operations (FLOPs). Therefore, the computational cost of two or more methods can be compared based on their FLOPs. Hence, we have evaluated the computational cost of the proposed



	Learnable parameters	Floating-point operations
DFE	402	1920
APAE	3794	23168
MemAE	12061	11297

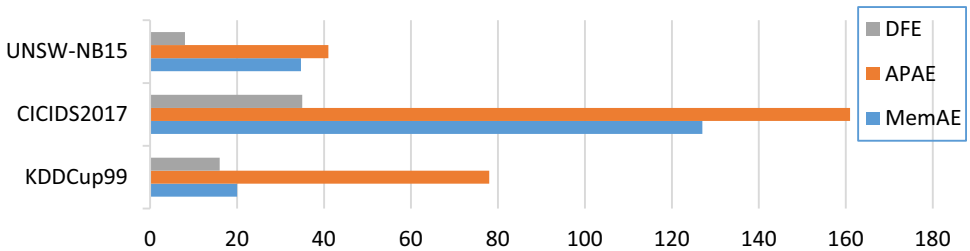
Fig. 19 Performance comparison results for multiclass classification on UNSW-NB15 dataset

architecture by comparing its FLOPs with state-of-the-art methods. The results have shown significant improvement over the state-of-the-art methods, and they are shown in Figs. 11, 12, 13, 15, 16, and 19. In addition, we have performed an empirical performance test for the proposed approach and other state-of-the-art methods. This section presents the results of this test, which is performed on all three datasets based on two criteria: the time for training the model with a batch of 128 records, and the time for inferring the class of a single record. Figure 20 shows the batch training time for DFE, APAE, and MemAE models on all three datasets. Note that the numbers are in the Seconds unit, and they are obtained on a virtual machine using a single-core Intel Xeon CPU with a 2.3 GHz clock. As you can see, the results confirm the significant superiority of the proposed approach over other methods in terms of batch training time. For all three datasets, the batch training time for the proposed approach is a fraction of the training time for other methods. For example, the training time for the proposed approach on the CICIDS2017 dataset is 35 s, which is about 21% and 27% of the training time for APAE and MemAE, respectively.

The second criterion of the empirical performance test is the inference time for a single record, which is in direct correlation with the number of FLOPs. It is also more important than the first criterion, as having a high training time for a model is not as bad as having a high inference time for that model. This is because that the training of a model can be done only once on a powerful machine, but, the inference should be repeatedly done in real-time on the target IoT device. Therefore, if a method has a high inference time, it may not be suitable for IoT devices. Figure 21 shows the single record inference time in milliseconds unit for DFE, APAE, and MemAE on all three datasets. As can be seen, the proposed method consistently performed better than the other two methods in all three datasets. The inference time for the proposed method in the KDDCup99 dataset is only 2.85 ms. This is while the APAE and MemAE inference times are 19.48 and 16.39 ms, respectively. This shows at least 82% improvement in inference time for the proposed method on this dataset compared to the other two methods. The situation for the other two datasets is also similar: the proposed method has at least 83% and 79% improvement in the inference time on the CICIDS2017 and UNSW-NB15 datasets, respectively.

4.7 Real-world usability comparison

Nowadays, IoT devices are small and cheap devices that have very limited resources. There are two main limitations



	KDDCup99	CICIDS2017	UNSW-NB15
DFE	16	35	8
APAE	78	161	41
MemAE	20	127	34.7

Fig. 20 Batch training time in seconds for DFE, APAE, and MemAE on all three datasets

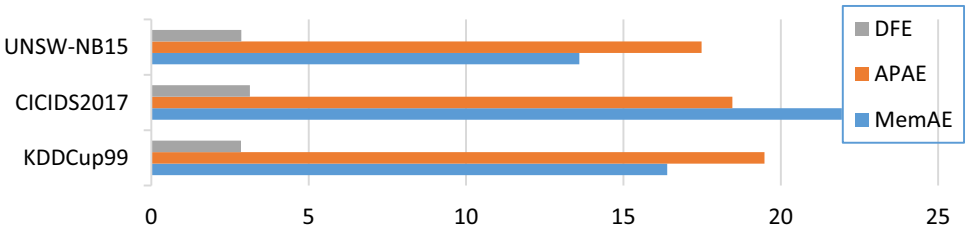


Fig. 21 Single record inference time in milliseconds for DFE, APAE, and MemAE on all three datasets

	KDDCup99	CICIDS2017	UNSW-NB15
DFE	2.85	3.13	2.86
APAE	19.48	18.46	17.48
MemAE	16.39	21.94	13.6

Table 7 Specification of three Arduino board for use as IoT devices [45]

	Processor	Main Memory
Arduino Leonardo	Atmel ATmega32u4 running at 16 MHz	2.5 KB
Arduino MKR NB 1500	ARM Cortex-M0 + CPU running at up to 48 MHz	32 KB
Arduino Due	Atmel AT91SAM3X8E running at 84 MHz	96 KB

for these devices: processing capability, and available main memory. For example, Table 7 shows the specification for three common Arduino boards that are currently being used as IoT devices. As you can see, these boards use slow processors and have very limited main memory.

In addition, in real-world applications, many of these devices run on batteries, which is why another limitation of these devices is electrical energy. Therefore, for an attack detection method to be usable on IoT devices, it should have low computational complexity, low power consumption, and low memory footprint. As described in the previous section, the proposed method is very computationally efficient compared to the previous works. Therefore, it is also more energy-efficient than previous works, as the energy consumption of a method depends heavily on its computational complexity. The third limitation, which is the main memory limitation, may be more important as it absolutely limits the usage of some algorithms on some devices. For example, Table 8 shows the memory usage of the proposed approach and the two other previous works APAE and MemAE, in the KB unit. As you can see, for all scenarios in Table 8, the proposed approach memory usage is about one-tenth and one-percent of the memory usage in the APAE and MemAE, respectively. This is very important as it allows using the proposed approach on all of the mentioned devices in Table 7. Even on the Arduino Leonardo, which has the lowest amount of main memory among the three boards, the proposed approach only uses at most 16% of the available memory. On the other hand, it is not possible to use APAE and MemAE in this board as the memory usage of these methods is more than the total available memory on this board. Also, the MemAE algorithm cannot be used in Arduino MKR NB 1500 because it consumes more memory than the total available memory on this board.

Considering the three limiting factors of computational complexity, power consumption, and memory usage, it is clear that the proposed approach is highly superior to the current state-of-the-art methods, and it is more suitable for use in real-world IoT applications.

Table 8 Memory usage for the proposed approach and two previous works on three different datasets in KB unit

	Binary classification			Multiclass classification		
	DFE	APAE	MemAE	DFE	APAE	MemAE
CICIDS2017	0.27	3.19	33.1	0.35	3.51	33.37
UNSW-NB15	0.26	3.19	33.19	0.39	3.70	33.48
KDDCup99	0.26	3.19	33.19	0.30	3.38	33.30

5 Discussion

The results of the experiments in the previous section showed that the gap between the overall classification accuracy of the proposed approach and the overall classification accuracy of previous works is small. However, for the comparison of intrusion detection systems, the overall classification accuracy is not the only important criterion. The efficiency of these systems is also very important, particularly in the IoT world, in which the hardware devices have very limited processing capabilities. An

intrusion detection system may have very good overall classification accuracy, but at the same time, it may need many processing resources, or it may have very bad classification performance in the minority classes. Therefore, it is necessary to look at various parameters while comparing different intrusion detection systems. Although the proposed method has slightly better overall classification accuracy than the previous works, it has very good classification performance in the minority classes, and at the same time, it is very lightweight compared to all the previous works. It significantly outperforms all the previous works in terms of the number of parameters and floating-point operations, which makes it ideal for real-time attack detection in IoT networks.

6 conclusion and remarks

In this paper, a very accurate NIDS called DFE has been presented. It has used a very lightweight and efficient neural network exploiting the idea of Deep Feature Extraction. The input vector of the network in the proposed model has been permuted in a 3D space, and its individual values are brought close together. This allowed the model to extract highly discriminative features using a small number of layers without the need to use large 2D or 3D convolution filters. As a result, the network has achieved an accurate classification using a significantly small number of needed calculations. This made the DFE ideal for real-time intrusion detection by IoT devices with limited processing capabilities. The efficacy of the DFE had been evaluated using KDDCup99, CICIDS2017, and UNSW-NB15 datasets and the results showed the superiority of the proposed model over state of the art algorithms. In the case of the KDDCup99 dataset, the proposed approach has provided almost the same overall classification accuracy as APAE and MemAE. It has outperformed the APAE and MemAE algorithms by reducing the number of required floating-point operations by about 92% and 81% respectively. For the CICIDS2017 dataset, the proposed method accuracy was about 1% higher than MemAE, while it has beat the APAE and MemAE algorithms in terms of the number of floating-point operations by about 91% and 85%, respectively. Finally, the results of the evaluation on the UNSW-NB15 dataset have shown that the proposed method has outperformed MemAE in terms of classification accuracy by at least 1.5%. In addition, the proposed method has surpassed the APAE and MemAE algorithms in terms of the number of floating-point operations by about 92% and 83%, respectively. However, there is still room for improvement in the proposed approach. The proposed approach is slightly weak compared to the APAE in terms of the classification of minority classes. This can be seen as future work, which can be done by modeling the problem of finding the best permutation as an optimization problem, and solving it using an optimization algorithm.

Funding Not applicable.

Declarations

Conflicts of interest Not applicable.

References

1. Fan J, Zhang Y, Wen W, Gu S, Lu X, Guo X (2021) The future of Internet of Things in agriculture: plant high-throughput phenotypic platform. J Clean Prod 280:123651
2. Philip NY, Rodrigues JJ, Wang H, Fong SJ, Chen J (2021) Internet of Things for in-home health monitoring systems: current advances, challenges and future directions. IEEE J Sel Areas Commun 39(2):300–310
3. Oniani S, Marques G, Barnovi S, Pires IM, Bhoi AK (2021) Artificial intelligence for internet of things and enhanced medical systems. In: Bhoi Akash Kumar, Mallick Pradeep Kumar, Liu Chuan-Ming, Balas Valentina E (eds) Bio-inspired Neurocomputing. Springer Singapore, Singapore, pp 43–59. https://doi.org/10.1007/978-981-15-5495-7_3
4. Gopikumar S, Raja S, Robinson YH, Shanmuganathan V, Chang H, Rho S (2021) A method of landfill leachate management using internet of things for sustainable smart city development. Sustain Cities Soc 66:102521
5. Sohi SM, Seifert J-, Ganji F (2021) RNNIDS: Enhancing network intrusion detection systems through deep learning. Comp Secur 102:102151
6. Sahar N, Mishra R, Kalam S (2021) Deep learning approach-based network intrusion detection system for fog-assisted IoT. In: Tiwari S, Suryani E, Ng AK, Mishra KK, Singh N (eds) Proceedings of international conference on big data, machine learning and their applications: ICBMA 2019. Springer Singapore, Singapore, pp 39–50. https://doi.org/10.1007/978-981-15-8377-3_4
7. Banadaki YM, Brook J, Sharifi S (2021) "Design of the network intrusion detection systems for the internet of things infrastructure using machine learning algorithms," in NDE 40 and smart structures for industry, smart cities, communication, and energy. Int Soc Opt Photon 11594:115940J
8. Wang F, Yang N, Shakeel M, Saravanan V (2021) Machine learning for mobile network payment security evaluation system. Trans Emerg Telecommun Technol. https://doi.org/10.1002/ett.4226
9. 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. Trans Emerg Telecommun Technol 32(1):e4150. https://doi.org/10.1002/ett.4150
10. Jeong S, Jeon B, Chung B, Kim HK (2021) Convolutional neural network-based intrusion detection system for AVTP streams in automotive Ethernet-based networks. Vehicular Commun 29:100338
11. Ji DJ, Park J, Cho D-H (2019) ConvAE: A new channel autoencoder based on convolutional layers and residual connections. IEEE Commun Lett 23(10):1769–1772

12. Wang Z, Zeng Y, Liu Y, Li D (2021) Deep belief network integrating improved kernel-based extreme learning machine for network intrusion detection. IEEE Access 9:16062–16091
13. Süzen AA (2021) Developing a multi-level intrusion detection system using hybrid-DBN. J Ambient Intell Humaniz Comput 12(2):1913–1923
14. Bilski J, Rutkowski L, Smoląg J, Tao D (2021) A novel method for speed training acceleration of recurrent neural networks. Inf Sci 553:266–279
15. Ma B, Jiang Z, Lu NL, Jiang Z (2020) Cybersecurity named entity recognition using bidirectional long short-term memory with conditional random fields. Tsinghua Sci Technol 26(3):259–265
16. Yuan S, Wu X (2021) Deep learning for insider threat detection: review, challenges and opportunities. Comp Secur 104:102221
17. Sharma N, Panwar D (2021) Advance security and challenges with intelligent IoT Devices. In: Goyal D, Chaturvedi P, Nagar AK, Purohit SD (eds) Proceedings of second international conference on smart energy and communication: ICSEC 2020. Springer Singapore, Singapore, pp 177–189. https://doi.org/10.1007/978-981-15-6707-0_17
18. Li T, Wu B, Yang Y, Fan Y, Zhang Y, Liu W. 2019 Compressing convolutional neural networks via factorized convolutional filters. InProceedings of the IEEE/CVF Conference on computer vision and pattern recognition (pp 3977-3986)
19. Deng L, Li G, Han S, Shi L, Xie Y (2020) Model compression and hardware acceleration for neural networks: a comprehensive survey. Proc IEEE 108(4):485–532. https://doi.org/10.1109/JPROC.2020.2976475
20. Basati A, Faghih MM (2021) APAE: an IoT intrusion detection system using asymmetric parallel auto-encoder. Neural Comput Appl. https://doi.org/10.1007/s00521-021-06011-9
21. Xin Y et al (2018) Machine learning and deep learning methods for cybersecurity. IEEE Access 6:35365–35381. https://doi.org/10.1109/ACCESS.2018.2836950
22. Tripathi G, Singh K, Vishwakarma DK (2019) Convolutional neural networks for crowd behaviour analysis: a survey. Vis Comput 35(5):753–776. https://doi.org/10.1007/s00371-018-1499-5
23. Alaeddine H, Jihene M (2021) Deep network in network. Neural Comput Appl 33:1453–1465
24. Vijayan M, Raguraman, and R. Mohan, (2021) A fully residual convolutional neural network for background subtraction. Pattern Recogn Lett 146:63–69. https://doi.org/10.1016/j.patrec.2021.02.017
25. Lv L, Wang W, Zhang Z, Liu X (2020) A novel intrusion detection system based on an optimal hybrid kernel extreme learning machine. Knowl-based Syst 195:105648
26. Zhang J, Ling Y, Fu X, Yang X, Xiong G, Zhang R (2020) Model of the intrusion detection system based on the integration of spatial-temporal features. Comp Secur 89:101681
27. Tian Q, Li J, Liu H (2019) A method for guaranteeing wireless communication based on a combination of deep and shallow learning. IEEE Access 7:38688–38695
28. Agarap AFM, A neural network architecture combining gated recurrent unit (gru) and support vector machine (SVM) for intrusion detection in network traffic data," presented at the Proceedings of the 2018 10th international conference on machine learning and computing, Macau, China, 2018. [Online]. Available: https://doi.org/10.1145/3195106.3195117
29. Zhou Y, Cheng G, Jiang S, Dai M (2020) Building an efficient intrusion detection system based on feature selection and ensemble classifier. Comput Netw 174:107247
30. Singh A, Kaur GS, Aujla RS, Batth, and S. Kanhere, (2020) DaaS: dew computing as a service for intelligent intrusion detection in edge-of-things ecosystem. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2020.3029248
31. Li X, Chen W, Zhang Q, Wu L (2020) Building auto-encoder intrusion detection system based on random forest feature selection. Comput Secur 95:101851
32. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell 2(1):41–50. https://doi.org/10.1109/TETCI.2017.2772792
33. Injadat M, Moubayed A, Nassif AB, Shami A (2020) Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans Netw Serv Manage. https://doi.org/10.1109/TNSM.2020.3014929
34. Gong D, Liu L, Le V, Saha B, Mansour MR, Venkatesh S, Hengel AV. 2019 Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. In: proceedings of the ieee/cvf international conference on computer vision (pp 1705-1714).
35. Miah MO, Khan SS, Shatabda S, Farid DM (2019) Improving detection accuracy for imbalanced network intrusion classification using cluster-based under-sampling with random forests, in 2019 1st international conference on advances in science, engineering and robotics technology (ICASERT), 1–5, doi: https://doi.org/10.1109/ICASERT.2019.8934495.
36. Roy AG, Navab N, Wachinger C (2018) Recalibrating fully convolutional networks with spatial and channel "squeeze and excitation" blocks. IEEE Trans Med Imaging 38(2):540–549
37. Tang J, Sun D, Liu S, Gaudiot J-L (2017) Enabling deep learning on IoT devices. Computer 50(10):92–96
38. Gong LLD, Le V, Saha B, Mansour MR, Venkatesh S, Van Den Hengel A, (2019) Memorizing normality to detect anomaly: memory-augmented deep autoencoder for unsupervised anomaly detection, in IEEE/CVF International conference on computer vision (ICCV), 1705–1714, doi: https://doi.org/10.1109/ICCV.2019.00179
39. Andresini G, Appice A, Di Mauro N, Loglisci C, Malerba D (2020) Multi-channel deep feature learning for intrusion detection. IEEE Access 8:53346–53359
40. Muhammad G, Hossain MS, Garg S (2020) Stacked autoencoder-based intrusion detection system to combat financial fraudulent. IEEE Internet Things J. https://doi.org/10.1109/JIOT.2020.3041184
41. Peng Y, Zhang L, Liu S, Wu X, Zhang Y, Wang X (2019) Dilated residual networks with symmetric skip connection for image denoising. Neurocomputing 345:67–76
42. Yao H, Fu D, Zhang ML, Liu Y (2019) MSML: a novel multi-level semi-supervised machine learning framework for intrusion detection system. IEEE Internet Things J 6(2):1949–1959. https://doi.org/10.1109/JIOT.2018.2873125
43. Al-Garadi MA, Mohamed A, Al-Ali AK, Du X, Ali I, Guizani M (2020) A survey of machine and deep learning methods for internet of things (IoT) security. IEEE Commun Surv Tutorials 22(3):1646–1685
44. Moustafa N, Slay J (2015) UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set), in 2015 Military communications and information systems conference (MilCIS), 1–6, doi: https://doi.org/10.1109/MilCIS.2015.7348942.
45. "Arduino Website." https://www.arduicc. Accessed 2021/11/1.