Neural Computing and Applications (2023) 35:4813–4833
https://doi.org/10.1007/s00521-021-06011-9

APAE: an IoT intrusion detection system using asymmetric parallel auto-encoder

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

Received: 31 December 2020 / Accepted: 1 April 2021 / Published online: 29 April 2021

Abstract
In recent years, the world has dramatically moved toward using the internet of things (IoT), and the IoT has become a hot research field. Among various aspects of IoT, real-time cyber-threat protection is one of the most crucial elements due to the increasing number of cyber-attacks. However, current IoT devices often offer minimal security features and are vulnerable to cyber-attacks. Therefore, it is crucial to develop tools to detect such attacks in real time. This paper presents a new and intelligent network intrusion detection system named APAE that is based on an asymmetric parallel auto-encoder and is able to detect various attacks in IoT networks. The encoder part of 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 proposed APAE approach has a lightweight and suitable architecture for real-time attack detection and provides very good generalization performance even after training using very limited training records. The efficacy of the APAE has been evaluated using three popular public datasets named UNSW-NB15, CICIDS2017, and KDDCup99, and the results showed the superiority of the proposed model over the state-of-the-art algorithms.

Keywords Internet of things · IoT · UNSW-NB15 · KDD · CICIDS2017 · IDS · Intrusion detection · Asymmetric auto-encoder · Convolutional neural networks · Attention module

1 Introduction

Despite the dramatically growing popularity of IoT networks [1, 2], the devices used in these networks are vulnerable to many security threats [3]. Therefore, the need for security in the context of IoT networks has become a prerequisite. To this end, the use of network intrusion detection systems (NIDS) has become a rational choice [4]. On the other hand, NIDSs need a lot of processing resources for inspection of the network flow to be able to detect attacks in real-time. However, current IoT devices have very limited processing capabilities, and the processing resources are very scarce in IoT networks. Therefore, real-time attack detection in IoT networks is a challenging task for NIDSs. In recent years, the high efficiency and accuracy of powerful deep learning algorithms dramatically increased their application in NIDS [5–8] and various other fields [9]. However, deep learning-based methods need a lot of processing resources, and therefore, they are not suitable for real-time detection of attacks in IoT networks. In addition, almost all of them have weak attack detection results in situations that there are few training data, because they need lots of training data. Various datasets exist for training a NIDS [10–12], but almost all of them suffer from the lack of training records in at least some classes of attacks. Therefore, almost all previous NIDS have weak classification results on these minority classes even if they have excellent overall classification accuracy on the dataset. Although some

✉ 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

researchers [13] have tried to augment these classes with some generated records and trained their model using augmented datasets, the generated records are synthetic and very similar to existing records. Therefore, this technique biases the network and makes it over-fit on training data. In addition, this technique does not solve the main problem, and the problem is still there: whenever some minority class exist the current models fail to extract the required information for generalization and, therefore, have weak performance in attack detection.

Convolutional neural networks (CNNs) are a subset of deep learning models that became a major trend in many deep learning tasks, specifically in NIDS [14]. However, current CNN-based NIDSs require many processing resources and training data and they are not appropriate for real-time attack detection in IoT networks in which there are inherent limitations in energy, memory, and processing resources. Some methods were recently used to lower the complexity of a deep model, including convolution factorization [15] and network compression [16]. Although current NIDSs have used various methods to decrease their complexity and increase their classification performance, they still suffer from two weaknesses: the heavy need for processing resources and weak performance in minority classes. This paper presents a novel CNN-based NIDS named APAE that solves the problem of IoT real-time attack detection even in minority classes. It has a lightweight architecture that can be used in IoT devices with limited capabilities. In addition, it provides very good generalization performance in minority classes after training using very limited training records. It also has very high accuracy and a low error rate that makes it suitable for real-time detection of malicious network packets.

The main contributions of this paper are as follows:

- Novel use of 2D representation for the input vectors of intrusion detection neural network that, unlike popular 1D representations, brings the individual parameters of input vectors close together and allows the convolutional filters to extract discriminative features from minority classes.
- Novel use of dilated convolution filters that (unlike standard convolutional filters) allows the network to have good generalization performance on minority classes by aggressively extracting spatial information across the inputs with fewer layers.
- Novel asymmetric auto-encoder named APAE that provides unsupervised feature learning by combining channel and position attention modules with dilated and standard deep auto-encoders and achieves high classification accuracy while maintaining compact architecture with very few numbers of parameters.

The rest of the paper is organized as follows: Sect. 2 presents relevant background information and examines existing research. Section 3 describes the proposed approach that is subsequently evaluated using various experiments in Sect. 4. Finally, the paper concludes in Sect. 5.

2 Background

2.1 Deep learning

Deep learning is an advanced field among machine learning subjects. It recently becomes too popular because of its superiority over previous machine learning techniques. It has made machine learning closer to true artificial intelligence by modeling complex relationships using multiple hierarchical levels of abstraction.

2.1.1 Convolutional neural networks

Convolutional neural networks (CNNs) are a subset of deep neural networks. They are also known as space invariant artificial neural networks and are most commonly used in analyzing visual images because of their translation invariance characteristics and shared-weight architecture [17]. They are inspired by the visual cortex of the brain in which individual neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. CNNs have a major advantage over traditional neural networks: unlike traditional neural networks that need human effort in preprocessing and feature design, CNNs need little preprocessing and achieve independence from prior knowledge by learning the convolutional filters as part of network trainable parameters.

CNNs typically have multiple successive layers, and the type of each layer is one of these three types: convolution layer, pooling layer, and fully connected layer [18]. The convolution layer is the core building block of CNNs. It has a set of kernels as learnable weights and convolves each kernel with its input to produce the output of the layer as a set of stacked activation maps. Formally, the kth activation map of a convolution layer can be represented as shown in Eq. (1):

$$h^k = f\left(X \otimes W^k + b^k\right) \tag{1}$$

where W^k corresponds to the weight vector, X is the input vector of the convolution layer, and b^k represents the bias. The function f represents the activation function that can be of different types such as rectified linear unit (ReLU), hyperbolic tangent function, sigmoid, and so on. The symbol $\otimes$ also indicates the convolution operation that can be formally represented as shown in equation

$$(X \otimes W)(p) = \sum_{s+t=p} X(s)W(t) \quad (2)$$

2.1.2 Auto-encoder

Auto-encoder (AE) is an unsupervised neural network [19]. Figure 1 shows a typical structure of an AE. The aim of training an AE is to achieve a network that produces its output layer as close to its input layer as possible. In other words, AE learns the best parameters to approximate the identity function and reconstruct its output as its input.

The auto-encoder has been used in anomaly detection by many researchers [20–22]. The idea is to use an AE that has been trained by some data representing normal states of a system. Having such an AE, a new unseen state of the system can be classified as the normal or anomaly as following: feeding AE with the new state as input, the new state is classified as normal if the AE can accurately reconstruct it in its output. On the other hand, the new state is classified as an anomaly if the reconstruction error is high (above some threshold).

Another way of using auto-encoder in intrusion detection has also been used in some researches [23]. Here, the idea is to use the auto-encoder as a dimensionality reduction tool. To explain that, the AE can be divided into two main parts: encoder and decoder. In general, the encoder transforms the input data into a lower-dimensional space, and then, the decoder expands it to reproduce the initial input data. The training procedure forces the AE to capture the most prominent features of the data distribution as a lower-dimensional representation at the heart of the hidden layers. Ideally, this lower-dimensional feature vector provides a better representation of the data points than the raw data itself and can be used as a compressed feature vector for classification.

2.2 Related works

Using a lightweight neural network, reference [24] proposed an IDS for resource-constrained mobile devices. The model, which is called IMPACT, uses a stacked auto-encoder (SAE) to reduce the length of the feature vector, besides using an SVM classifier for the detection of only impersonation attacks.

Reference [25] uses a deep neural network named FC-Net to detect attacks when only a few malicious samples are available for training. The method performs two operations: feature extraction and comparison. It learns a pair of feature maps from a pair of network traffic samples for classification and then compares the resulting feature maps to infer whether that pair of samples belongs to a particular class or not.

Reference [26] presents a variational auto-encoder-based method for intrusion detection. This method is an anomaly detection approach that detects network attacks using reconstruction error of auto-encoder as an anomaly score. However, like any other anomaly detection method, this method is only capable of distinguishing between good and bad packets and cannot detect different attack types.

Another hybrid anomaly detection system is proposed in reference [27]. This system also has two stages: the first stage uses a classical auto-encoder to extract features from the input vector, and the second stage applies a deep neural network to the output of the first stage with the aim of anomaly detection.

In reference [28], a two-stage deep learning model is used as an intrusion detection system that includes a stacked auto-encoder and a soft-max classifier. The first stage is to detect normal or abnormal packets, and in the second stage, the method classifies between normal and other classes of attacks. The authors reported a good classification accuracy on KDD99 and UNSW-NB15 datasets, but their method is not suitable for IoT networks due to the need for powerful processing resources.

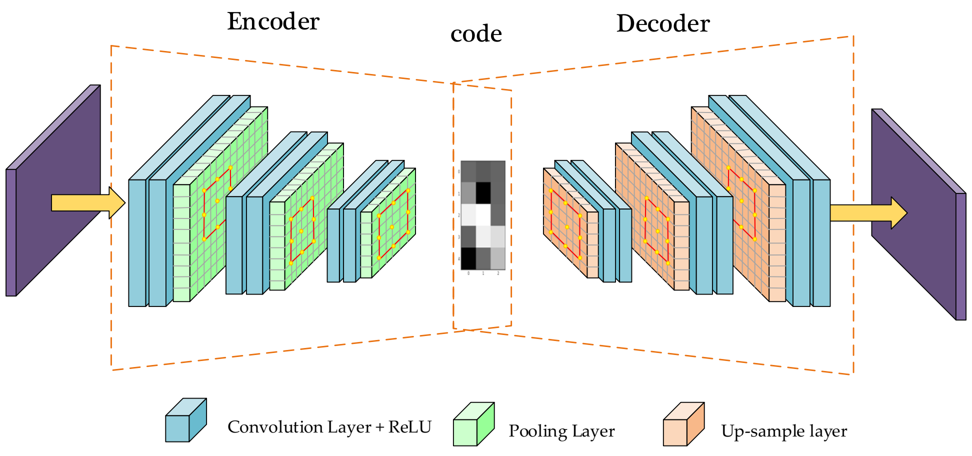

Fig. 1 A typical structure for an auto-encoder

Yet another auto-encoder-based method is presented in reference [29]. It is named the SAVAER-DNN method and uses Wasserstein GAN with gradient penalty (WGAN-GP) to augment the training data. It synthesizes samples of low-frequent and unknown attacks, and as a result, it has better accuracy in the detection of low-frequent attacks.

In reference [30], an anomaly detection algorithm is presented that uses an auto-encoder together with a random forest. It also uses feature selection and feature grouping to decrease training time. The feature grouping also helps in increasing the accuracy of the trained model by placing the features with similar orders of magnitude into the same group.

Reference [31] 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 error, leading to miss-detection of anomalies. 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 [32] proposed a multi-layer classification approach to identify minority-class intrusions in highly imbalanced data. The method first classifies incoming 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 [33] introduced a collaborative intrusion detection system-based deep blockchain network for the identification of cyber-attacks in IoT networks. The paper also proposed a hybrid privacy-preserving method to provide confidentiality of smart contracts by combining blockchain with a trusted executed environment. The authors have evaluated their BiLSTM based intrusion detection method on the UNSW-NB15 dataset and showed that it has good accuracy and detection rate in classifying attack events exploiting cloud networks.

Reference [34] used deep learning together with shallow learning. It proposes to use a non-symmetric deep auto-encoder (NDAE) for non-symmetric data dimensionality reduction alongside a random forest for classification. The model has a light architecture and archives good classification accuracy on KDDCup99 and NSL-KDD datasets. Due to its lightweight architecture, this method is a good candidate for doing intrusion detection in IoT networks.

Reference [35] presented an intrusion detection framework named MSML (multilevel semi-supervised machine learning) that includes four modules: pure cluster extraction, pattern discovery, fine-grained classification, and model updating. The pure cluster module tries to find all pure clusters using a hierarchical semi-supervised k-means algorithm. The aim of the pattern discovery module is to find unknown patterns and to label the test samples as either labeled known patterns or unlabeled unknown patterns. The fine-grained classification module determines the class for unknown pattern samples, and the model-updating module provides a mechanism for retraining. The method is evaluated using the KDDCup99 dataset, and the evaluation results showed that the MSML could provide good accuracy and F1-score.

An IoT ensemble IDS is presented in [36] that mitigates particular botnet attacks against DNS, HTTP, and MQTT protocols. In this work, new statistical flow features from the protocols are generated using the UNSW-NB15 dataset, and an AdaBoost learning method is developed using three machine-learning techniques: decision tree, naive Bayes, and artificial neural network. The authors have evaluated their ensemble technique and showed that the generated features have the potential characteristics of malicious and normal activity using the correntropy and correlation coefficient measures.

Reference [37] proposed an intrusion detection system based on a stacked auto-encoder and a deep neural network. In this work, the stacked auto-encoder decreases the dimensionality of the input vector in an unsupervised manner, and the deep neural network is trained in a supervised manner to extract deep-learned features for the classifier. The neural network in this paper has two or three layers of which each one contains a fully connected, a batch normalization layer, and a dropout layer. The authors have evaluated their proposed method using the KDDCup99 and two other datasets and have reported the results of binary and multiclass classification for their work.

A dew computing as a service (DaaS) for improving the performance of intrusion detection in edge of things (EoT) ecosystems has been proposed in [38]. 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.

Although various state-of-the-art NIDSs exist, there are still some technical gaps among them, and almost all of existing NIDSs suffer from the weak performance in minority classes and the heavy need for processing resources. Therefore, to fill these gaps, better NIDSs are still needed that can give better classification results in minority classes. To this end, this paper presents a novel NIDS named APAE that has a lightweight architecture, and provides very good classification performance in minority classes even after training using very limited training records.

3 Proposed approach

This section presents various aspects of the proposed approach. In the following, Sect. 3.1 explains the idea behind using 2D data representation. Section 3.2 presents the dilated convolution concept and compares it with normal convolution. In Sect. 0, the proposed idea of using asymmetric parallel auto-encoder has been explained, and finally, Sect. 3.5 describes the proposed classification method for network intrusion detection.

3.1 Preprocessing and input data representation

The aim of NIDS is to detect intrusion by monitoring data obtained from network traffic. Therefore, NIDSs usually capture network traffic and, based on some 1-dimensional feature vector extracted from network packets, decide whether to raise the intrusion alarm or not. This feature vector usually includes parameters like protocol type, service type, number of failed logins, and so on. Current public datasets like KDDCup'99 [11] and CICIDS2017 [10] are also of this form. To the best of our knowledge, all deep learning-based state-of-the-art NIDSs also use 1D feature vectors as their network input. However, this 1D representation creates an obstacle to achieving high classification accuracy with convolutional neural networks. To explain that, extracted 1D feature vectors from network packets are not like voice or any other type of signal. In the classification of voice signals, the class of a signal correlates with extracted information from local changes of consecutive points as well as the global shape of the signal. Therefore, the order of the points in a voice signal (or any other type of signal) is fixed and cannot be changed. However, the order of individual parameters in the input vector of a CNN-based NIDS is important and can highly affect the network complexity and classification accuracy. Consider A and B as two individual parameters in the input vector of a CNN, and if the class of an input vector has a correlation with the information extracted from the relation between values of A and B, to achieve a simple and accurate CNN, it is important for A and B to be as close as possible. This is because the convolution filters can only extract information from neighboring parameters in the feature vector, and if A and B be far from each other in the input vector, larger filters or more layers of filters are needed to extract this information.

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, as shown in Fig. 2, a 2D filter can extract more neighborhood information than a 1D filter. Although it is possible to add more layers to the network or increase the size of the 1D filter in order to extract correlation information from far-away parameters, this increases the network complexity and decreases the accuracy of the network due to the need for more training data and time.

For example, to extract the relative information between the values of x_6 and x_{18} of the 1D feature vector in Fig. 2a, a filter of size 1×13 is needed while this information can be extracted from the 2d feature vector of Fig. 2b using a 2d filter of the size 3×3. For this reason, the proposed approach uses a 2D representation of feature vectors as network input. In the preprocessing phase, the proposed approach transforms the feature vectors to their 2D representation with equal width and height while padding the end of them 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 Dilated convolution

Standard convolution filters (Eq. 2) can extract spatial information only from neighboring parameters. Therefore, to extract spatial information from distant parameters, larger filters or more successive layers of filters are required that makes the network more complex. To overcome this problem, dilated convolution [39] can be used as a generalized form of convolution. It adds one hyper-parameter to the convolution filter called dilation. Figure 3 shows the difference between standard convolution and dilated convolutions. As it can be seen, the dilation parameter represents the space between neighboring parameters. In other words, an L-dilated convolution filter considers cells with the distance of L as neighbors and extracts their relative spatial information.

Equation (1) shows the formal definition of dilated convolution. As you can see, dilated convolution is a generalization of standard convolution, and choosing the

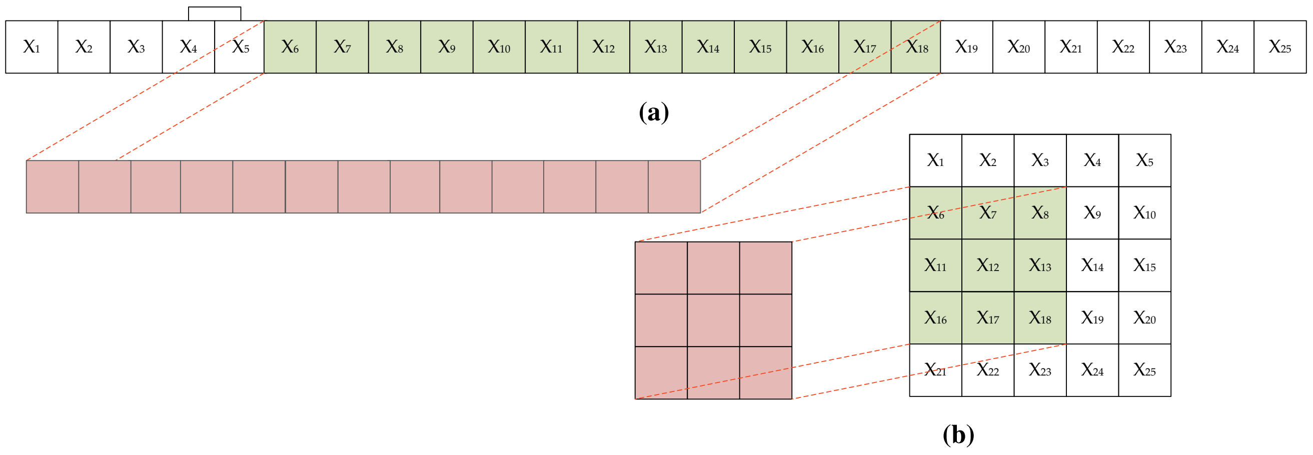

Fig. 2. 1D feature vector versus 2D feature vector

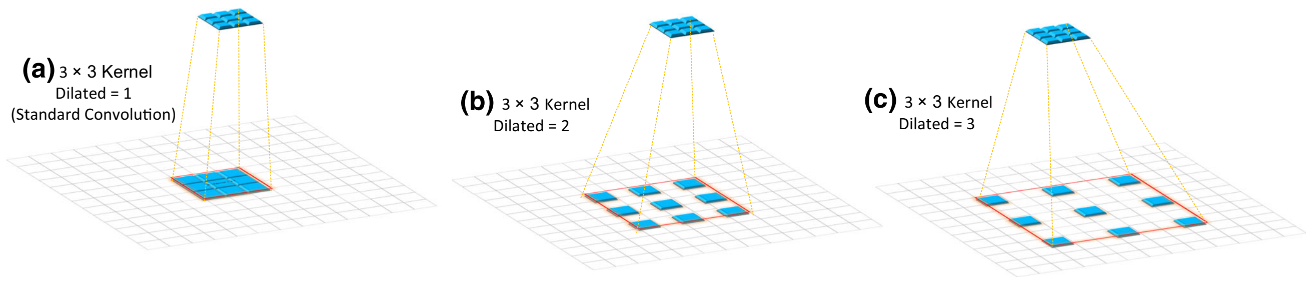

Fig. 3 Dilated convolution versus standard convolution

value of $L = 1$, makes Eq. (1) equal to the equation of standard convolution (Eq. 2).

$$(X \otimes_L W)(p) = \sum_{s+Lt=p} X(s)W(t) \quad (3)$$

In the case of the proposed approach, dilated convolution is very useful because it allows the proposed approach to extract relative spatial information across the inputs much more aggressively and with fewer layers.

3.3 Attention modules

Convolution operations have a local receptive field; therefore, the obtained features by a convolution operation may have some differences for input vectors with the same. These differences introduce intra-class discrepancy and affect the classification accuracy. To overcome this issue, it is necessary to obtain global contextual information from input feature vectors by building associations among long-range features. Attention modules can do this. An attention module maps a query and a set of key-value pairs to an output, where the query, keys, values, and output are all matrices [40]. The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key. A special type of attention module is the self-attention module in which all of the keys, values, and queries come from the same place. The self-attention module can extract the global contextual information from long-range input features, and hence, the proposed approach uses two types of self-attention modules (provided in [41]) as explained in the next two subsections.

3.3.1 Positional self-attention module

It has been shown that local features obtained by convolutional filters lead to misclassification [41]. Therefore, long-range contextual information is necessary for discriminant feature representation. To this end, the proposed approach uses the positional self-attention module (Fig. 4) to enhance the representation capability of local features by encoding a wider range of contextual information into them.

As shown in Fig. 4, the positional self-attention module takes a local feature $\mathbf{A} \in \mathbb{R}^{C \times H \times W}$ as its input and feeds it into a convolution layer to generate two new feature maps $\mathbf{B}$ and $\mathbf{C}$ where $\{\mathbf{B}, \mathbf{C}\} \in \mathbb{R}^{C \times H \times W}$ (C, H, and W are, respectively, the number of channels, height, and width of the input). Then, $\mathbf{B}$ and $\mathbf{C}$ are reshaped to $\mathbb{R}^{C \times N}$ where $N = H \times W$ and a softmax layer is applied on the result of multiplication of $\mathbf{B}$ and transpose of $\mathbf{C}$ to obtain the

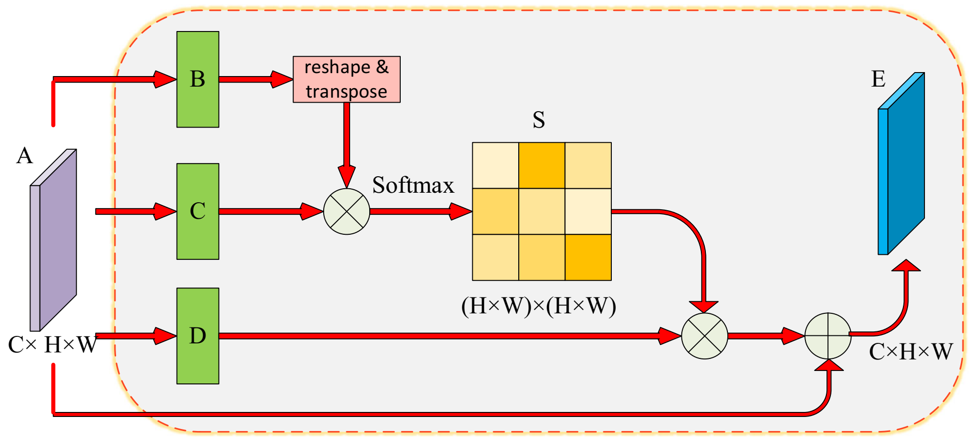

Fig. 4 The structure of positional self-attention module

attention map $\mathbf{S} \in \mathbb{R}^{N \times N}$. This way, each value of S_{ij} measures the ith position impact on the jth position and can be obtained by Eq. (4):

$$S_{ij} = \frac{e^{B_i \cdot C_i}}{\sum_{i=1}^{N} e^{B_i \cdot C_i}} \tag{4}$$

In the meantime, a new feature map $\mathbf{D} \in \mathbb{R}^{C \times N}$ is generated by applying a convolution layer on the $\mathbf{A}$ and reshaping the result to $\mathbb{R}^{C \times N}$. The output $\mathrm{E} \in \mathbb{R}^{C \times H \times W}$ of the position attention module is then obtained by reshaping the result of the following formula to $\mathbb{R}^{C \times H \times W}$:

$$E_j = \alpha \sum_{i=1}^{N} \left(S_{ij} D_{ij}\right) + A_j \tag{5}$$

Here, α is a learnable scaling parameter. Equation (3) shows that $\mathbf{E}$ at each position is equal to a weighted sum of the features across all positions and original features. Therefore, it has a global contextual view and can improve intra-class consistency by selectively aggregating contexts according to the spatial attention map [41].

3.3.2 Channel self-attention module

Each channel map of input features can be seen as a class-specific response, and different semantic responses relate to one another. Therefore, the interdependencies between channel maps can be used to emphasize interdependent feature maps and improve the representation capability of features. Hence, the proposed approach uses a channel self-attention module (shown in Fig. 5) to explicitly model interdependencies between channels.

As illustrated in Fig. 5, the channel attention map $\mathrm{X} \in \mathbb{R}^{C \times C}$ is directly calculated from the original input features $\mathbf{A}$. First, $\mathbf{A}$ is reshaped to $\mathbb{R}^{C \times N}$, and then, $\mathbf{X}$ is obtained by applying a softmax layer on the multiplication of $\mathbf{A}$ with its transpose as in Eq. (4):

$$X_{ij} = \frac{e^{A_i \cdot A_i}}{\sum_{i=1}^{N} e^{A_i \cdot A_i}} \tag{6}$$

Here, each X_{ij} measures the impact of the ith channel on the jth channel, and the final output $\mathbf{E} \in \mathbb{R}^{C \times H \times W}$ can be obtained by reshaping the result of multiplying a learnable scaling factor β by the element-wise sum of the outcome of multiplication of $\mathbf{A}$ and the transpose of $\mathbf{X}$ as illustrated in Eq. (5):

$$E_j = \beta \sum_{i=1}^{C} \left(X_{ij} D A_{ij}\right) + A_j \tag{7}$$

From Eq. (7), it can be inferred that each final value of the channel attention module is a weighted sum of the features of all channels and the original features and can be used to model the long-range semantic dependencies between feature maps.

3.4 Asymmetric parallel auto-encoder

A deep auto-encoder can be used to reduce the dimension of the input vector using the encoder-decoder paradigm described in Sect. 2.1.2. This way, one can extract the most descriptive features for classification and discard deceptive information in the input vector that misleads the classifier. However, standard deep auto-encoders have their limitations in NIDS. They use standard convolution filters, and therefore, to achieve good accuracy in extracting discriminative features, multiple numbers of them should be stacked together. This results in complex deep networks that are not suitable for IoT devices. In addition, standard auto-encoders are symmetrical, i.e., the encoder and decoder have an equal number of layers, and each layer in the decoder exactly does the reverse operations of its corresponding layer in the encoder as shown in Fig. 1. However, as explained in [42], having more units in fewer layers of the encoder and meanwhile having more layers in the decoder helps the auto-encoder to achieve a better

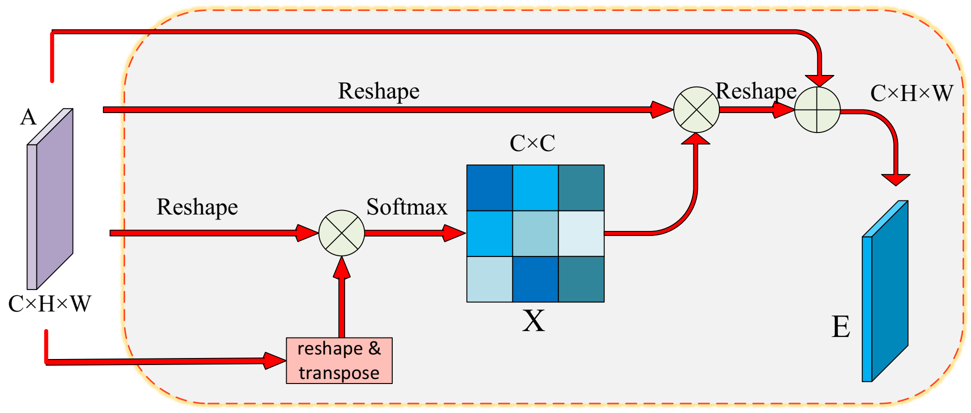

Fig. 5 The structure of channel self-attention module

reconstruction of the input data. The more units in fewer layers of encoder allow finding a robust abstract representation, while the more layers of decoder help it to reconstruct the input data from the abstract representation. Therefore, the proposed approach uses an advanced auto-encoder called asymmetric parallel auto-encoder (APAE). Figure 6 shows the overall structure of an APAE. As you can see, APAE is an asymmetric auto-encoder that combines two encoders and a decoder together. The input to APAE is the 2D representation of the input vector obtained by applying the preprocessing procedure described in Sect. 3.1. Generally, APAE consists of a transfer layer and three other major parts: encoder, latent feature, and decoder. The transfer layer has eight standard convolution filters transferring the input to eight representation channels. These channels then split into two equal parts to feed the encoder.

The encoder contains two feature extractors: a long-range feature extractor and a local feature extractor. The long-range feature extractor is the encoder part of an auto-encoder with dilated convolutional filters of size 3 × 3. The local feature extractor is also the encoder part of an auto-encoder with standard 3 × 3 convolutional filters. In each feature extractor, there are three layers of successive convolutional filters that each one (together with down-sampling), extracts some lower dimension features from its input and feed them to the next layer. The final parts of the two feature extractors are also different. The local feature extractor has a positional self-attention module, and the long-range feature extractor has a channel self-attention module. Using a positional self-attention module as the last

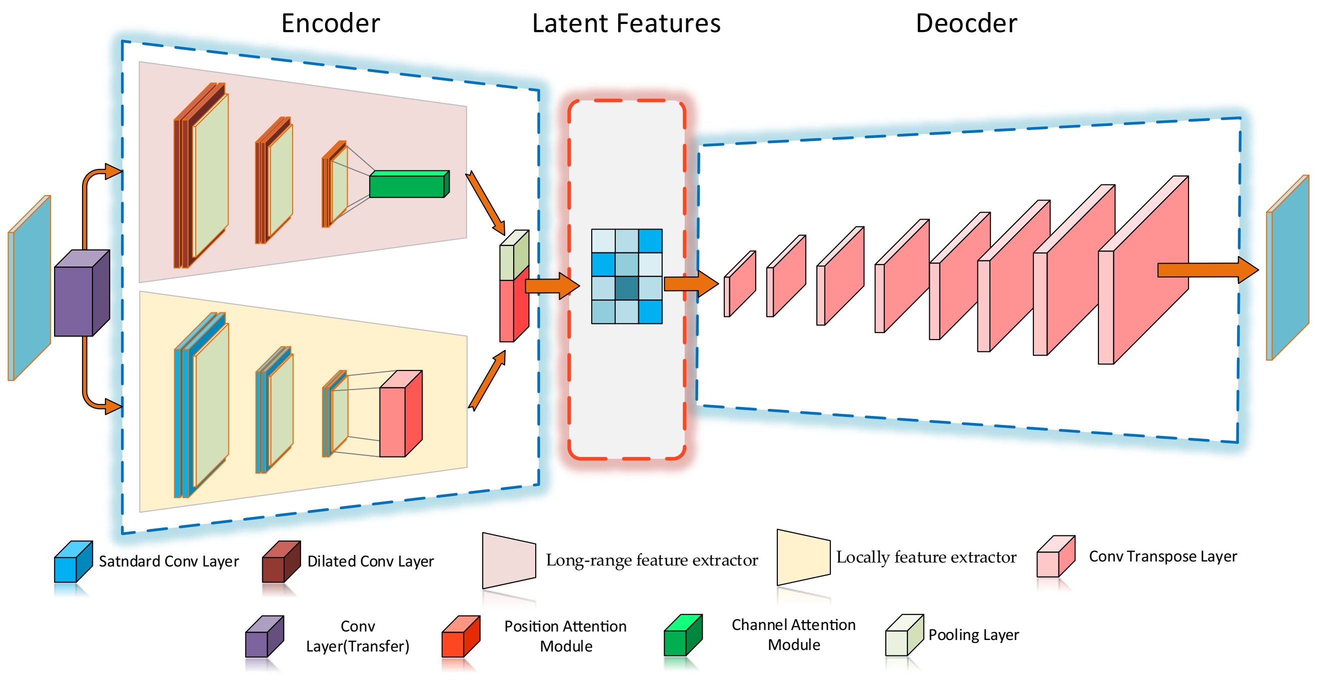

Fig. 6 The overall structure of an asymmetric parallel auto-encoder

part of the local feature extractor is because the positional self-attention module can enhance the representation capability of local features, as explained in Sect. 3.3.1. The channel self-attention module, as the last part of the long-range feature extractor, is to emphasize interdependent feature maps and to enhance the extracted long-range features, as described in Sect. 3.3.2. At the end of each feature extractor, there are compressed features with lower dimensions that together make the latent features that are used to feed the decoder.

Typically, a symmetric auto-encoder uses convolutional layers together with pooling layers in the encoder to reduce the dimensionality of the inputs and obtain a lower-dimensional representation. The decoder part also uses up-sampling layers (the reverse of pooling layer) together with convolutional layers to regenerate the original inputs using the output of the encoder. However, as explained in previous paragraphs, the APAE encoder generates the lower-dimensional representation of input data by combining the outputs of two parallel feature extractors that each one uses a different type of convolutional layers. Therefore, the APAE decoder should be able to reverse the operation of both dilated and standard convolution layers. To this end, the APAE uses a robust decoder containing transposed convolution layers. Unlike the up-sampling layer that has no learnable parameter, the transposed convolution has learnable parameters and, on its own, can do the job of both up-sampling and convolution layers. By stacking eight transposed convolution layers, the APAE decoder can effectively regenerate the APAE inputs with high accuracy (as shown in the evaluation results in Sect. 4).

Like normal auto-encoders, the aim of training an APAE is to approximate the identity function. In addition, the latent features of a trained APAE can also be used as a compressed and lower dimension form of its inputs. However, the difference between APAE and normal auto-encoders is that the APAE is more robust and can efficiently produce the lower dimension representation of its input with a smaller number of layers and parameters. In addition, APAE is more suitable for NIDS because it acts more aggressively in extracting the relative spatial information of its input parameters comparing to normal auto-encoders.

3.5 Network intrusion detection

The proposed approach is a deep learning model for detecting intrusion in IoT networks. Because IoT devices have limited processing resources, the model should be as lightweight and with few parameters as possible. To this end, the proposed model uses the encoder part of an APAE (outlined in the previous section) as an efficient tool to extract the most descriptive low dimension features for accurate classification.

In the proposed model, an APAE is first trained on a dataset to estimate the identity function for the training data. Then, the final model (as shown in Fig. 7) is obtained by concatenating a fully connected classifier layer at the end of the first parts (transfer layer, encoder, and latent features) of the trained APAE. After that, the final model is trained again using the training data to obtain the classifier weights while fine-tuning the weights of the APAE encoder for accurate classification.

Note that, in deep learning research, the success of a model is dictated by its structural architecture. Existing deep learning approaches typically use 1-dimensional input vector and they also have sequential structures that use only standard convolution layers in sequence. Therefore, they need to have many layers to achieve good classification accuracy, and as a result, they require many processing resources, which make them not suitable for real-time attack detection in IoT networks. In contrast, the structure of the proposed model has resulted from rational decisions together with experimenting with several structural compositions to achieve the best results. The proposed model uses a parallel structure that have two feature extractors, each having few sequential layers with different types of convolutional filters. It also uses 2-dimensional input vector which brings the long-range features close together and allows the convolutional filters of the local feature extractor to extract more useful information. In addition, the long-range feature extractor uses dilated convolutional filters that can extract more information from long-range features. The proposed model also uses a positional self-attention module to enhance the representation capability of local feature extractor, and a channel self-attention module to enhance the extracted long-range features by the long-range feature extractor. Therefore, as the results of experiments in the next section shows, the proposed model is highly superior comparing to the current state of the art NIDSs as it has an efficient and lightweight architecture with very few parameters and also, it can accurately discern normal network traffic and different classes of attacks (even minority classes) from each other.

4 Results and experiments

The proposed model is implemented with python using the Pytorch library. The Adam optimizer with a learning rate equal to 0.001 has been used to train the proposed model in 40 epochs on Google Colab infrastructure. This section presents the results of experiments and compares the proposed approach with the works presented in [33–38, 43, 44] using three public datasets: CICIDS2017, UNSW-NB15,

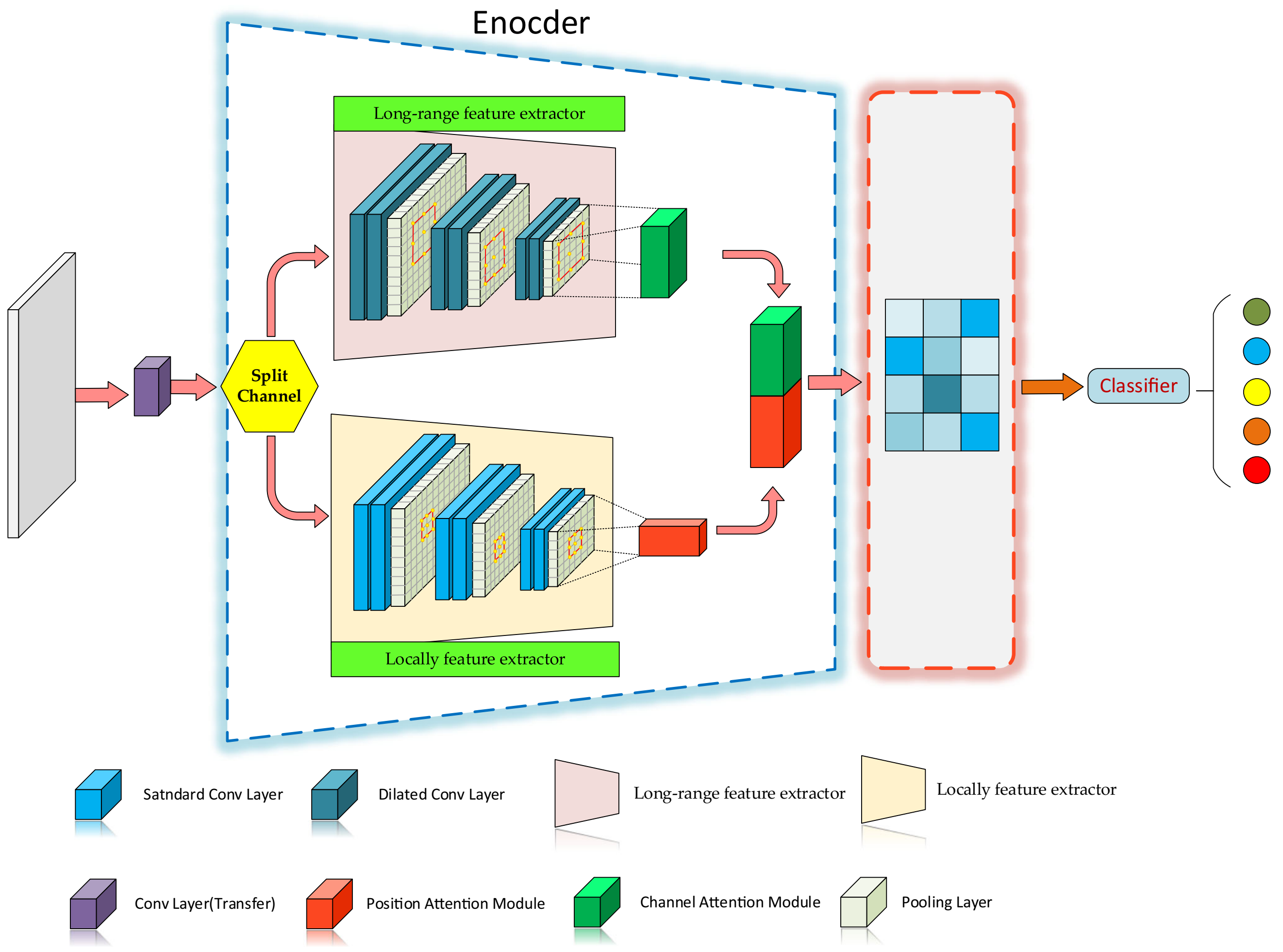
Enocder
Long-range feature extractor
Split Channel
Locally feature extractor
Classifier
Satndard Conv Layer
Dilated Conv Layer
Long-range feature extractor
Locally feature extractor
Conv Layer(Transfer)
Position Attention Module
Channel Attention Module
Pooling Layer

Fig. 7 The proposed model: network intrusion detection using an APAE together with a classifier

and KDDCup99. Two types of experiments have been done on each dataset. In the first type of experiment (Sect. 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 (multi-class 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 (NDAE [34] and MemAE [44]) made the source code of their works publicly available (NDAE source code [45], MemAE source code [46]). Therefore, we used these source codes to obtain the required results for the NDAE and MemAE 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 corresponding paper.

4.1 Datasets

4.1.1 NSW-NB15

The UNSW-NB15 dataset [12, 47] 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 be compatible with the results of previous works. Among the records in this dataset, 63.9% belongs 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 multi-class and binary labels. From the remaining 47 features, 42 features are numeric, and five features are categorical. After applying the preprocessing that is explained in Sect. 3.1, each record of this dataset can be represented by a 2D matrix of the size 8 × 8.

4.1.2 CICIDS2017

The CICIDS2017 dataset [10] is one of the most popular databases 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 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, and Brute force) and 14 types of sub-attacks. 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. 3.2, 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.

4.1.3 KDDCup99

KDDCup99 dataset is a known benchmark dataset in IDS research [35, 37]. 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 that each one represents a single connection record with 41 features, including both 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. Each vector 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 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 Comparison criteria

To justly compare the efficiency and accuracy of the proposed model with other NIDSs, the following four criteria have been used:

1. Overall classification accuracy:

Considering N as the total number of records in the test set and T as the total number of correct classifications made by a classifier, the overall accuracy of the classification (shown in equation (6)) can be calculated as the ratio between T and N multiplied by 100.

$$\text{accuracy} = \frac{T}{N} \times 100 \tag{8}$$

Confusion matrix:

A confusion matrix is a squared table that allows visualization of the performance of a classification algorithm. It makes it easy to see if the algorithm confused between two classes and mislabeled each one as another. In this table, each row and column represent a class. The value of a cell at the index (i, j) shows the number of data instances that actually belong to the ith class, but the algorithm predicted them as members of the jth class. It is easy to visually inspect the prediction errors using this table as all the correct predictions are on the diagonal of the table, and the values outside the diagonal represent the prediction errors.

Precision, recall, and F-Score:

These criteria are useful for class-wise evaluation of the output of the classifier. To define these criteria, the following definitions are also needed for each class c:

- True positive (TP): number of records that are properly classified into the class c.
- False positive (FP): number of records that are mistakenly classified into the class c.

- False negative (FN): number of records from class c that are mistakenly classified into other classes.

Having previous definitions, the precision, recall, and F score for each class can be obtained using the following formulas:

$$\text{Precision} = \frac{TP}{TP + FP} \tag{9}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{10}$$

$$F - \text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} \ + \text{Recall}} \tag{11}$$

The F score is the harmonic mean of the recall and the precision. The highest possible value of the F score is 1, which indicates perfect precision and recall, and the lowest possible value is 0, if either the precision or the recall is zero. In the case of classes with uneven distribution, the F score is a better criterion comparing to the *accuracy*.

The Number of parameters:

Since IoT devices have very limited processing capabilities, it is important for an IoT NIDS to be as computationally efficient as possible. Therefore, a highly CPU intensive NIDS is not suitable for IoT networks even if it provides a perfect classification of different attack types. From this point of view, a better NIDS for IoT is the one that has low computational complexity while providing good classification results. The number of parameters is a good criterion to measure the computational complexity of deep learning methods, and therefore, it is used in this paper to compare the proposed approach (which is a deep learning method) with other methods.

4.3 Anomaly detection

This section presents the results of the anomaly detection experiment. In this experiment, the records of all attack types are combined into a single attack class for each dataset, and each record of the three datasets is labeled as either Normal or Attack. Section 4.3.1 shows the results of this experiment on the UNSW-NB15 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 KDDCup99 dataset.

4.3.1 UNSW-NB15 dataset

The anomaly detection results of the proposed approach against the other algorithms using the UNSW-NB15 dataset are shown in Table 1. 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. Reference [43] 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.

The proposed approach is highly superior to both NDAE and MemAE in terms of efficiency and performance. The proposed model has only 1162 parameters that show about 65% and 90% decrease in network complexity comparing to NDAE and MemAE, respectively. This is very important as the proposed model can be used in devices with low processing power, and hence, it better suits to IoT networks comparing to other NIDSs.

4.3.2 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. 4.1.1. The results on this dataset also confirm the superiority of the proposed approach over almost all other methods. Again, the classification accuracy is higher than the NDAE and MemAE. The proposed method achieved about 4% superiority over the NDAE method while its accuracy is a little higher than the MemAE algorithm and is slightly lower than the accuracy of the presented method in reference [43].

In the case of performance, the proposed method is again highly superior to the other two methods. The number of parameters of the proposed method is about 12% and 77% lower than the number of parameters in NDAE and MemAE algorithms, respectively, which makes the proposed method to be more suitable for anomaly detection in IoT networks comparing 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 contains the remaining 35% of the data and has been used as the test set for the evaluation of all algorithms.

4.3.3 KDDCup99 dataset

This section evaluates the anomaly detection performance of the proposed APAE approach against other algorithms using the KDDCup99 dataset outlined in Sect. 4.1.3. The results obtained from anomaly detection on the KDDCup99 dataset by APAE and the other algorithms are presented in Table 3. By comparing the results of all algorithms, it is obvious that the accuracy of the proposed model is better than other algorithms. In terms of classification accuracy,

Table 1 The results of anomaly detection for UNSW-NB15 dataset

	Precision (%)			Recall (%)			*F* score (%)		
	NDAE	MemAE	APAE	NDAE	MemAE	APAE	NDAE	MemAE	APAE
Normal	99.92	99.99	**100**	99.92	100	**100**	99.92	99.99	**100**
Attack	99.94	100	**100**	99.94	99.99	**100**	99.94	100	**100**

	APAE	MemAE	NDAE	[43]	[35]	[37]	[38]	[33]	[36]
	Overall performance								
Accuracy (%)	**100**	99.99	99.92	100	N/A	N/A	N/A	N/A	N/A
Total parameters	**1162**	11,621	3372	N/A	N/A	N/A	N/A	N/A	N/A

The values with the bold text are the ones that are maximum of their correponding rows in the sub-tables (Precision, Recall, and F-Score)

Table 2 The results of anomaly detection for CICIDS2017 dataset

	Precision (%)			Recall (%)			*F* score (%)		
	NDAE	MemAE	APAE	NDAE	MemAE	APAE	NDAE	MemAE	APAE
Normal	98.57	**99.70**	99.38	93.99	96.20	**98.04**	96.23	97.92	**98.71**
Attack	94.45	96.46	**98.13**	98.69	**99.72**	99.41	96.52	98.06	**98.77**

	APAE	MemAE	NDAE	[43]	[35]	[37]	[38]	[33]	[36]
	Overall performance								
Accuracy (%)	98.73	98.19	94.64	**99.9**	N/A	N/A	N/A	N/A	N/A
Total parameters	**3274**	14,621	3732	N/A	N/A	N/A	N/A	N/A	N/A

The values with the bold text are the ones that are maximum of their correponding rows in the sub-tables (Precision, Recall, and F-Score)

Table 3 The results of anomaly detection for KDDCup99 dataset

	Precision (%)			Recall (%)			*F* score (%)		
	NDAE	MemAE	APAE	NDAE	MemAE	APAE	NDAE	MemAE	APAE
Normal	95.37	99.30	**99.86**	98.86	99.91	**99.94**	97.09	**99.90**	99.86
Attack	99.72	99.98	**99.98**	98.83	99.83	**99.97**	99.27	99.97	**99.97**

	APAE	MemAE	NDAE	[37]	[35]	[43]	[38]	[33]	[36]
	Overall performance								
Accuracy (%)	**99.95**	99.84	98.83	93.8	N/A	N/A	N/A	N/A	N/A
Total parameters	3274	9996	**3177**	N/A	N/A	N/A	N/A	N/A	N/A

The values with the bold text are the ones that are maximum of their correponding rows in the sub-tables (Precision, Recall, and F-Score)

the proposed model is 1% better comparing to the NDAE algorithm, while it is almost comparable to the MemAE algorithm and is far better than the results of reference [37]. This is while the proposed approach is highly superior to the MemAE in terms of efficiency and performance. The proposed model has 3,274 parameters that show about a 67% decrease in network complexity comparing to MemAE. However, in terms of performance, NDAE beats the APAE by a small margin because APAE has 2% more parameters.

4.4 Multi-class classification

This section presents the results of the multi-class classification experiment. In this experiment, the algorithms are compared based on their ability to predict the true classes of dataset records. Section 4.4.1 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. The results for the

CDCIDS2017 dataset are also presented in Sect. 4.4.2, in which there are 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 for the KDDCup99 dataset that has five classes: a single class of normal records, beside four attack classes of U2R, Dos, R2L, and Probe.

4.4.1 UNSW-NB15 dataset

Table 4 shows the classification results for the 10-class attack detection experiment on the UNSW-NB15 dataset for the APAE against other algorithms. As you can see, the APAE defeats the NDAE and MemAE in almost all criteria for individual classes. It should be noted that the distribution of records in this dataset is unbalanced. For example, the Worms and Shellcode classes are minority comparing to other classes. The total number of records in the Exploits class is 44,525 records, while the Worms class has only 130 records in the train set and 44 records in the test set. The Shellcode class also contains only 1133 records in train set and 378 records in test set. Therefore, as it can be seen from Table 4, the NDAE and the MemAE have poor classification performance in these classes. This is while the APAE classification performance on these classes is very significant. These differences are better shown in Fig. 8. As it can be seen, the APAE precision and recall for all classes are higher than the other two algorithms. The NDAE has the worse results in almost all classes; its precision and recall for the Worms and Shellcode classes are equal to zero. Also, the NDAE precision and recall for the Exploits and Analysis classes are also very bad and almost equal to zero. The situation for the MemAE is better than the NDAE in the Exploits, Analysis, and Shellcodes 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 MemAE has high number of false negatives, and as a result, it has very low recall. The confusion matrixes of Fig. 9 are also show the NDAE incorrectly classified all the records of the Worms and Shellcodes classes. 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. 9 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.

In terms of the number of parameters, the APAE beats the MemAE and loses to NDAE. The number of parameters for APAE is about 68% lower than the MemAE. The number of parameters of NDAE is about 12% less than the APAE; however, the classification accuracy of APAE is fare better. Comparing the classification accuracy of APAE with other methods in Table 4 also shows that the APAE is superior to all of them.

Table 4 The results of multiclass classification for UNSW-NB15 dataset

	Precision (%)			Recall (%)			*F* score (%)		
	NDAE	MemAE	APAE	NDAE	MemAE	APAE	NDAE	MemAE	APAE
Normal	68.78	99.81	**99.96**	94.91	100	**100**	79.76	99.91	**99.98**
Reconnaissance	17.64	100	**100**	3.03	96.28	**99.80**	5.17	98.11	**99.90**
Backdoor	12.66	88.08	**98.31**	8.23	91.25	**100**	9.98	89.64	**99.15**
Dos	15	85.84	**99.46**	12.74	98.26	**99.56**	13.78	91.63	**99.51**
Exploit	2.21	98.91	**100**	0.04	93.70	**99.53**	0.07	96.24	**99.77**
Analysis	3.12	90.35	**96.98**	5.47	99.56	**99.56**	3.97	94.73	**98.25**
Fuzzers	16.07	**99.77**	99.69	16.58	98.14	**99.84**	16.32	98.94	**99.76**
Worms	0	8.19	**91.67**	0	84.09	**100**	0	14.92	**95.65**
Shellcode	0	52.78	**100**	0	5.03	**100**	0	9.18	**100**
Generic	98.84	99.89	**100**	95.95	99.88	**100**	97.37	99.89	**100**

	APAE	MemAE	NDAE	[38]	[33]	[36]	[35]	[37]	[43]
	Overall performance								
Accuracy (%)	**99.89**	98.23	66.73	85.71	99.41	99.5	N/A	N/A	N/A
Total parameters	**3794**	12,061	3372	N/A	N/A	N/A	N/A	N/A	N/A

The values with the bold text are the ones that are maximum of their correponding rows in the sub-tables (Precision, Recall, and F-Score)

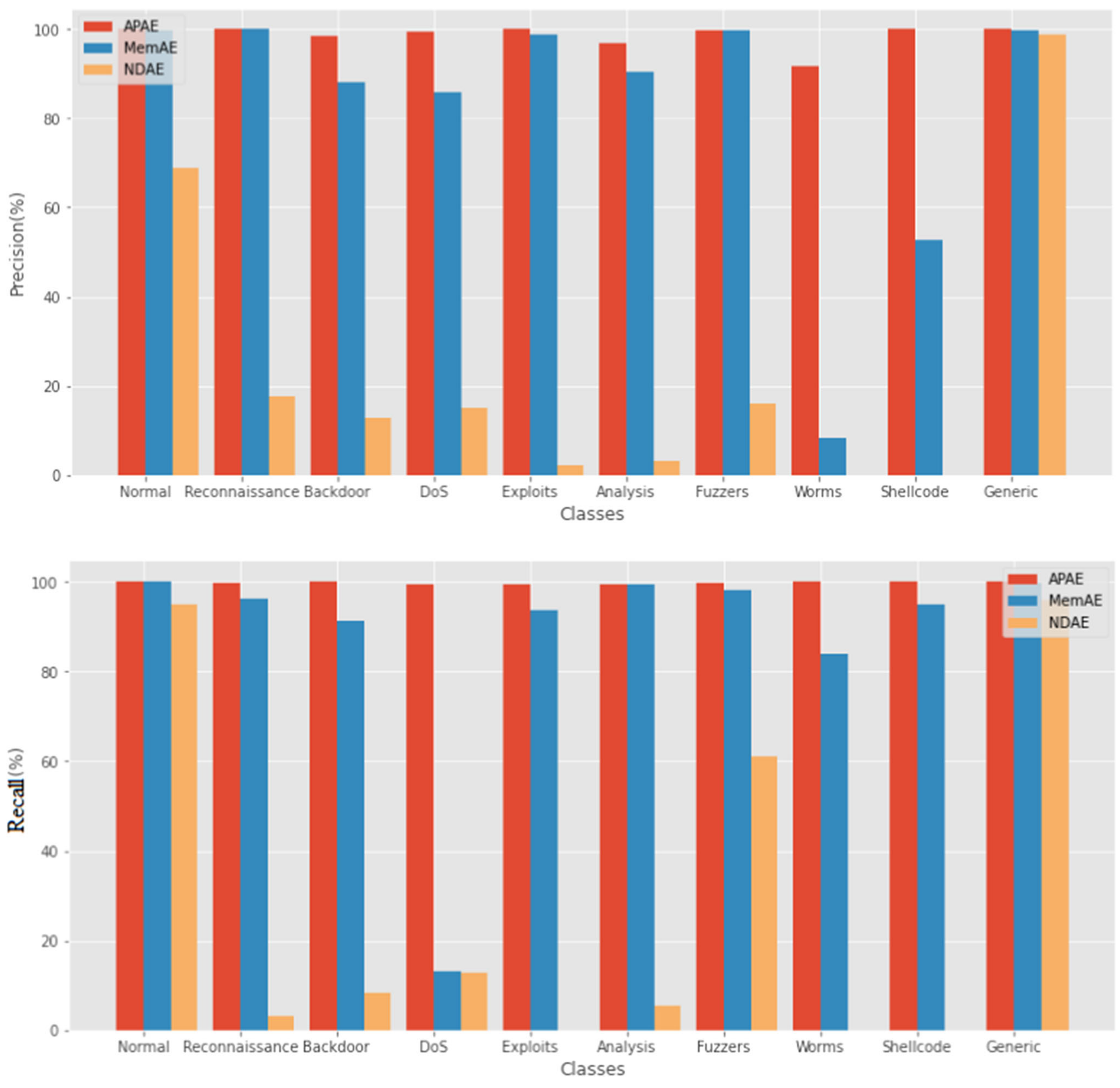

Fig. 8 Precision and recall comparison on UNSW-NB15 dataset

4.4.2 CICIDS2017 dataset

Table 5 shows the results of the 7-class attack detection experiment on the CICIDS2017 dataset for APAE against other algorithms. Note that references [33, 35–38, 43] did not evaluate their works on this dataset, and their source codes are also not publicly available. However, the source code for the MemAE and the NDAE algorithms are publicly available and we used their source codes to obtain the results of this experiment for these algorithms. Therefore, Table 5 only shows the results of this experiment for the APAE, the MemAE, and the NDAE algorithms, which their source codes are publicly available. As you can see in Table 5, the results show that the APAE has dominance over the other methods in almost all evaluated parameters. The APAE has achieved the overall accuracy of 99.5%, which is about 1% and 2% higher than MemAE and NDAE, respectively. This is while the APAE is highly efficient comparing to the MemAE algorithms, and the number of parameters in the APAE is significantly lower than the number of parameters in the MemAE algorithms. The APAE has only 3599 parameters, which shows 76% improvement in efficiency comparing to the MemAE that has 15,016 parameters. The APAE is also more efficient

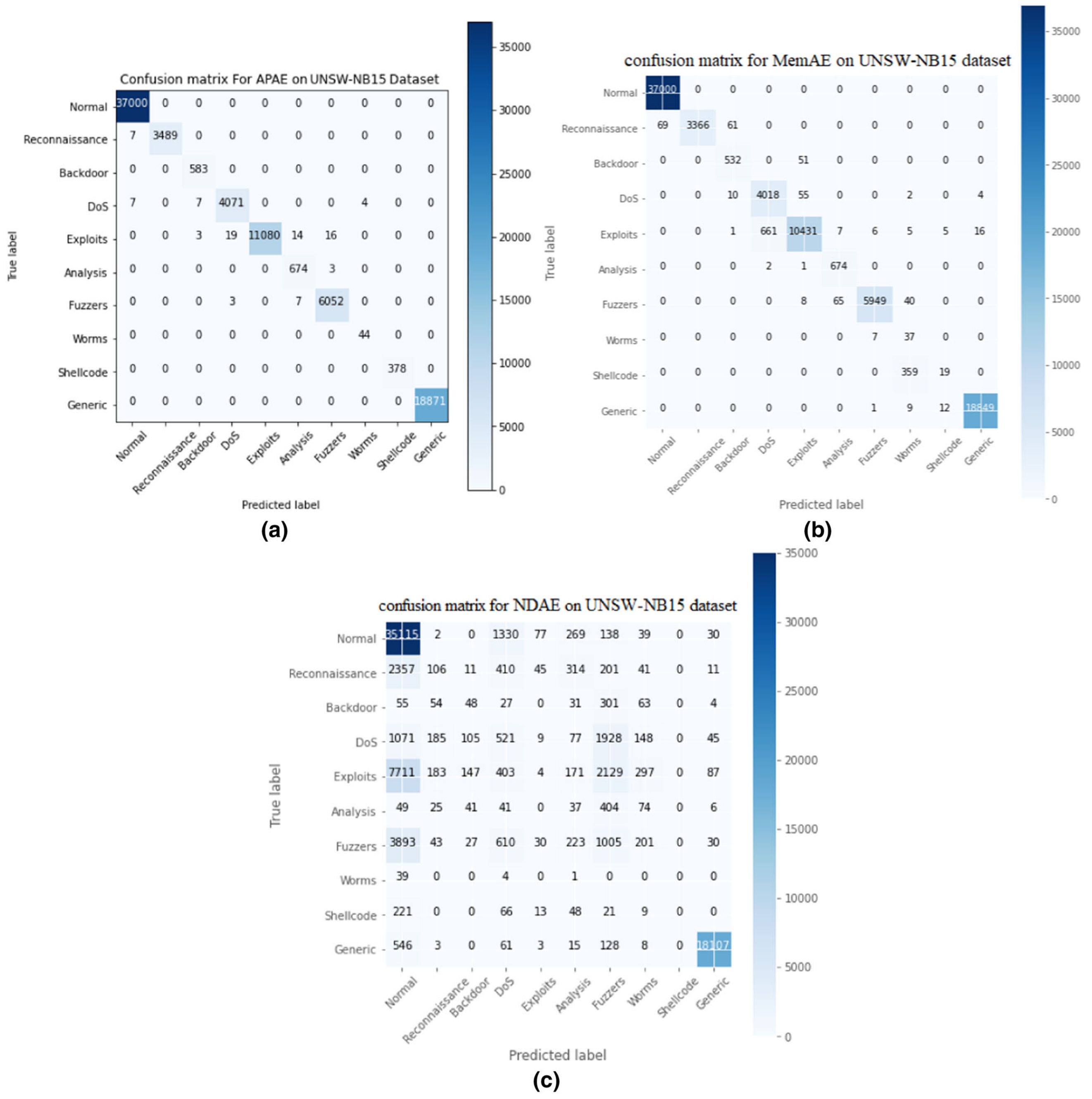

Fig. 9 Multiclass confusion matrix for **a** APAE, **b** MemAE, and **c** NDAE algorithms on UNSW-NB15 dataset

than the NDAE, and it has about 3% lower number of parameters than the NDAE, which has 3732 parameters. This again verifies the superior performance of the APAE and its true effectiveness for multi-class attack detection in IoT networks.

Almost all class-wise parameters for APAE are also higher than the other two algorithms. Nevertheless, the results for Infiltration class are substantial. 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, including the NDAE and the MemAE, 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 comparing to the other two methods. The precision and recall charts in Fig. 10 and confusion matrixes shown in Fig. 11 also confirm that both the NDAE and MemAE incorrectly classified all 12 records in the test set of

Table 5 The results of multiclass classification for CICIDS2017 dataset

	Precision (%)			Recall (%)			*F* score (%)		
	NDAE	MemAE	APAE	NDAE	MemAE	APAE	NDAE	MemAE	APAE
Benign	98.48	99.39	**99.81**	96.58	97.23	**99.21**	97.52	98.30	**99.51**
DoS	98.40	98.90	**99.47**	99.25	99.77	**99.84**	98.82	99.34	**99.65**
PortScan	93.42	93.99	**99.58**	98.03	99.83	**99.88**	95.67	96.82	**99.73**
Infiltration	0	0	**38.89**	0	0	**58.33**	0	0	**46.67**
Web Attack	56.74	**88.24**	64.37	53.73	9.48	**96.84**	55.19	17.12	**77.33**

	APAE	MemAE	NDAE	[35]	[37]	[43]	[38]	[33]	[36]
	Overall performance								
Accuracy (%)	**99.50**	98.26	97.56	N/A	N/A	N/A	N/A	N/A	N/A
Total parameters	**3599**	15,016	3732	N/A	N/A	N/A	N/A	N/A	N/A

The values with the bold text are the ones that are maximum of their correponding rows in the sub-tables (Precision, Recall, and F-Score)

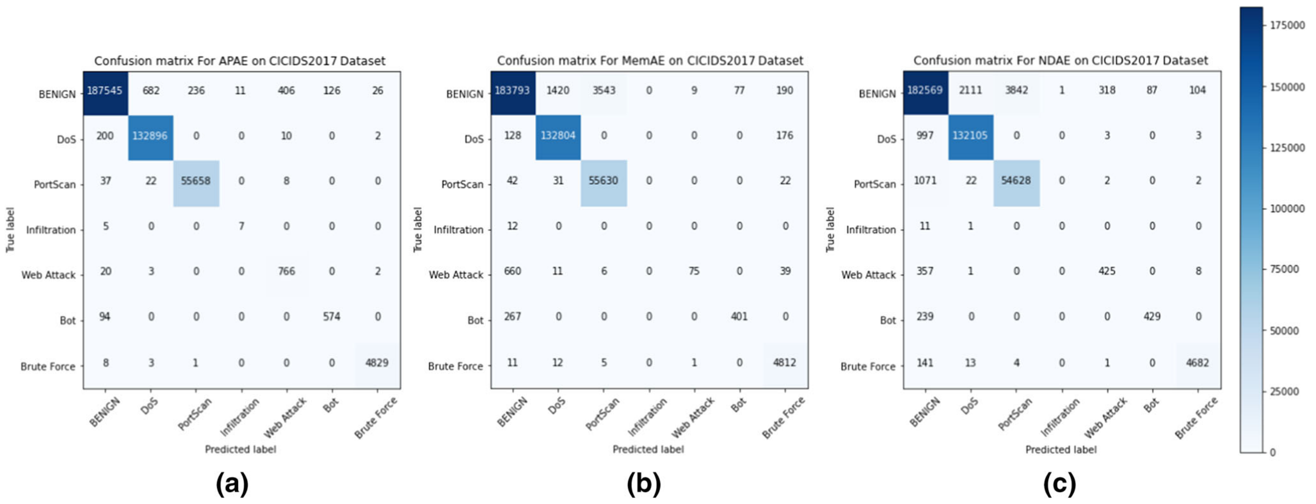

Fig. 10 Precision and recall comparison on CICIDS2017 dataset

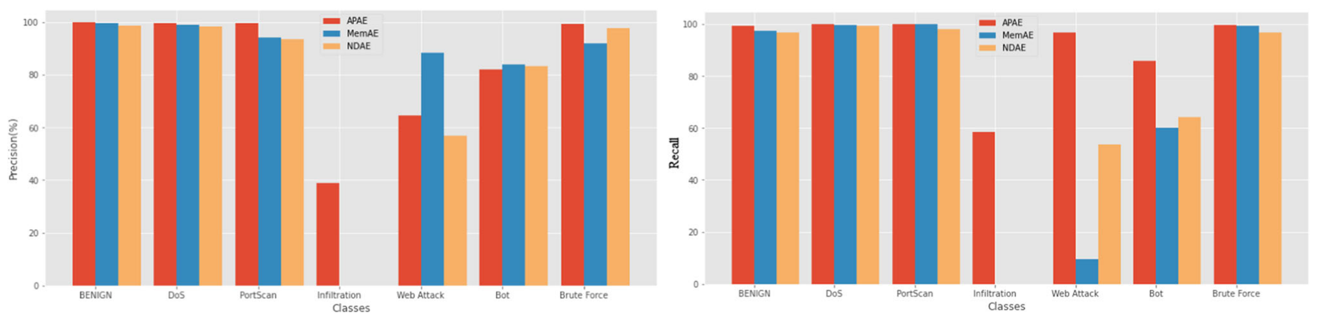

Fig. 11 Multiclass confusion matrix for **a** APAE, **b** MemAE, and **c** NDAE algorithms on CICIDS2017 dataset

Infiltration class, while the classification results of the proposed APAE are correct in 7 cases of total of 12 test records. The results for the Bot and Web Attack classes are also notable. In the Web Attack class, the MemAE has better precision comparing to the APAE. Both the NDAE and the MemAE also slightly outperform the APAE in terms of precision in the Bot class. However, in both of these classes, the recall for the APAE is significantly higher than the recall for both the other two algorithms. Also, the APAE beats the two other algorithms in terms of the

Table 6 The results of multiclass classification for KDDCup99 dataset

	Precision (%)			Recall (%)			*F* score (%)		
	NDAE	MemAE	APAE	NDAE	MemAE	APAE	NDAE	MemAE	APAE
Normal	94.36	99.66	**99.82**	99.02	99.51	**99.91**	96.63	99.58	**99.86**
U2R	0	0	**64.71**	0	0	**68.75**	0	0	**66.67**
DoS	99.19	99.99	**100**	99.18	100	**100**	99.19	100	**100**
R2L	62.77	70.0	**96.73**	19.93	89.86	**89.86**	30.26	78.70	**93.17**
Probe	40.24	**99.90**	98.95	3.14	96.48	**98.95**	5.83	98.16	**98.95**

	APAE	MemAE	NDAE	[35]	[37]	[43]	[38]	[33]	[36]
	Overall performance								
Accuracy (%)	**99.94**	99.83	98.13	96.6	94.2	N/A	N/A	N/A	N/A
Total parameters	3469	10,122	**3177**	N/A	N/A	N/A	N/A	N/A	N/A

The values with the bold text are the ones that are maximum of their correponding rows in the sub-tables (Precision, Recall, and F-Score)

F-Score, which is a better criterion in the case of classes with uneven distribution.

4.4.3 KDDCup99 dataset

Table 6 presents the results of the 5-class classification experiment on the KDDCup99 dataset for APAE, NDAE, MemAE, and other algorithms. The results show that the APAE is again superior to the other methods in almost all evaluated parameters. The proposed method achieved higher values in accuracy, precision, recall, and F-score for almost all five classes. However, the remarkable results are the ones obtained for the U2R class. Note that the data in the KDDCup99 dataset are also unevenly distributed in different classes. For example, the U2R class has merely 52 records at all, while the Dos class has 293582 training records. Therefore, almost no previous NIDSs have good classification performance on the U2R class. The NDAE and MemAE are also not exceptions, and as you can see in Table 6, their classification performance on the U2R class is very bad. However, the classification performance of the proposed APAE algorithm on the U2R class is very high comparing to the other methods. This is also observable from confusion matrixes shown in Fig. 12 and charts in Fig. 13: both the NDAE and MemAE incorrectly classified all 16 records in the test set of U2R class, while the classification results of the proposed APAE are correct in 11 cases of total 16 test records.

The overall performance results on this dataset also confirm the dominance of the proposed approach over the other methods in multi-class classification. The APAE accuracy is slightly higher than the MemAE algorithm, while it achieved about 1% superiority over the NDAE method in terms of accuracy. Its accuracy is also far better than the presented methods in reference [35, 37].

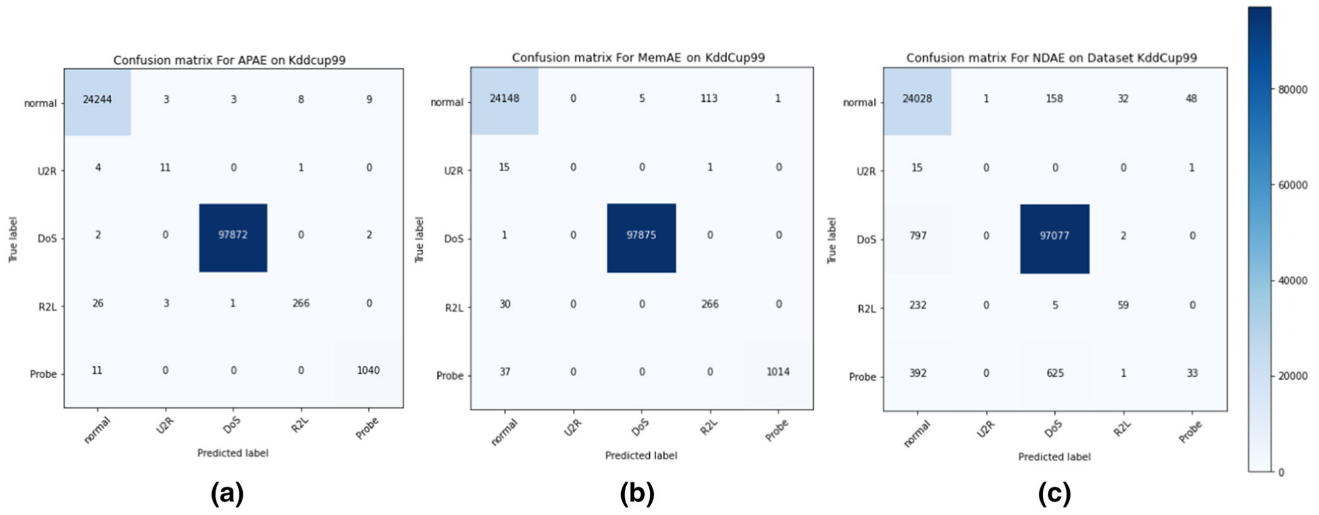

Fig. 12 Multiclass confusion matrix for **a** APAE, **b** MemAE and **c** NDAE algorithms on KDDCup99 dataset

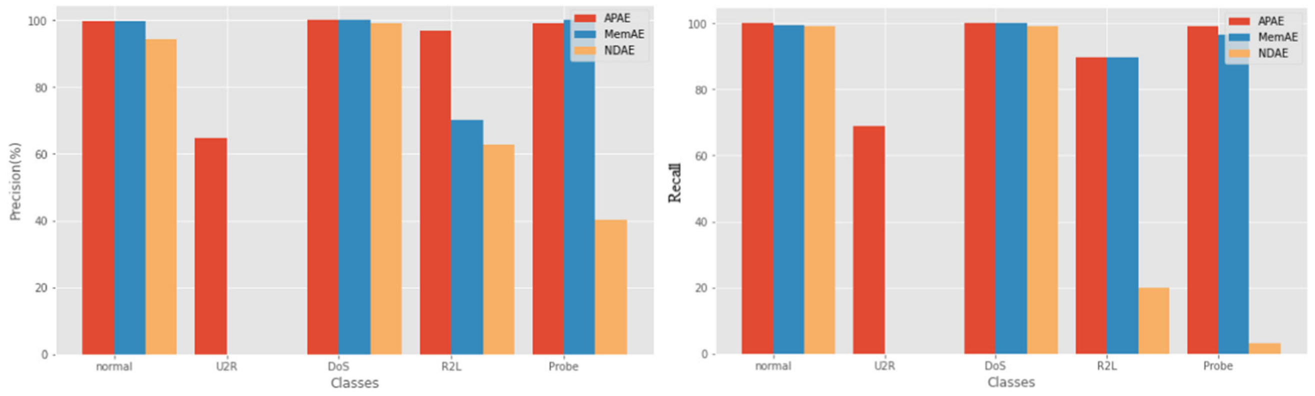

Fig. 13 Precision and recall comparison on KDDCup99 dataset

Comparing to MemAE, the APAE again has a smaller number of parameters. The number of APAE parameters is about 65% lower than the number of parameters in MemAE, which shows the advantage of the APAE in terms of performance and its true effectiveness for multi-class attack detection in IoT networks.

5 Discussion

As the results of the experiments in Sect. 4 showed, the gap between the overall classification accuracy of the previous works is small. The gap between the overall classification accuracy of the proposed approach and the overall classification accuracy of previous works is also 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. Another important criterion for comparing intrusion detection systems is the sub-class classification accuracy, especially in minority classes, as this shows the ability of the system in cases that the training samples are very limited. 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, the proposed approach is highly superior to the previous works in terms of the classification performance in the minority classes. Also, the proposed approach is very lightweight comparing to the previous works, because it has fewer numbers of parameters, as mentioned in the experimental results of Sect. 4.

6 Conclusion and remarks

In this paper, a new and lightweight architecture based on asymmetric parallel auto-encoder (APAE) has been proposed that has used dilated and standard convolutional filters to extract both locally and long-range information around individual values in the feature vector. It also has a positional self-attention and a channel self-attention module to enhance the local and long-range features, respectively. This separation of features allowed the APAE to achieve an accurate attack detection (even in minority classes) with a small and lightweight deep neural network suitable for IoT devices with low processing capabilities. The effectiveness of the proposed neural network has been evaluated using UNSW-NB15, CICIDS2017, and KDDCup99 datasets, and the results showed the superiority of the proposed model over the state-of-the-art algorithms.

Funding None.

Availability of data and material None.

Declaration

Conflict of interest None.

References

1. Stoyanova M, Nikoloudakis Y, Panagiotakis S, Pallis E, Markakis EK (2020) A survey on the internet of things (IoT) forensics: challenges, approaches and open issues. IEEE Commun Surv Tutor 22:1191
2. Al-Garadi MA, Mohamed A, Al-Ali A, Du X, Ali I, Guizani M (2020) A survey of machine and deep learning methods for internet of things (IoT) security. IEEE Commun Surv Tutor 22:1646
3. Popoola SI, Adebisi B, Hammoudeh M, Gui G, Gacanin H (2020) Hybrid Deep Learning for Botnet Attack Detection in the Internet of Things Networks. IEEE Int Things J 8:4944

4. Mbarek B, Ge M, Pitner T (2020) Enhanced network intrusion detection system protocol for internet of things. In Proceedings of the 35th annual ACM symposium on applied computing, pp 1156–1163.
5. Louati F, Ktata FB (2020) A deep learning-based multi-agent system for intrusion detection. SN Appl Sci 2(4):1–13
6. Gao M, Song Y, Xin Y (2020) Intrusion detection based on fusing deep neural networks and transfer learning. In Digital TV and wireless multimedia communication: 16th international forum, IFTC 2019, Shanghai, China, September 19–20, 2019, Revised selected papers, Springer Nature, Berlin, vol 1181, p 212.
7. Rashid A, Siddique MJ, Ahmed SM (2020) Machine and deep learning based comparative analysis using hybrid approaches for intrusion detection system. In: 2020 3rd International conference on advancements in computational sciences (ICACS): IEEE, pp 1–9.
8. Gamal M, Abbas H, Sadek R (2020) Hybrid approach for improving intrusion detection based on deep learning and machine learning techniques. Joint European-US workshop on applications of invariance in computer vision. Springer, pp 225–236
9. D. Vallejo-Huanga (2020) Empirical exploration of machine learning techniques for detection of anomalies based on NIDS. IEEE Latin Am Trans 100(1e)
10. Intrusion Detection Evaluation Dataset (CIC-IDS2017): https://www.unb.ca/cic/datasets/ids-2017.html
11. KDD Cup 1999 Data: http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
12. The UNSW-NB15 Dataset: https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/
13. Araujo-Filho PFd, Kaddoum G, Campelo DR, Santos AG, Macêdo D, Zanchettin C (2020) Intrusion detection for cyber-physical systems using generative adversarial networks in fog environment. IEEE Int Things J 1–1. doi: https://doi.org/10.1109/JIOT.2020.3024800.
14. Chaabouni N, Mosbah M, Zemmari A, Sauvignac C, Faruki P (2019) Network intrusion detection for IoT security based on learning techniques. IEEE Commun Surv Tutor 21(3):2671–2701. https://doi.org/10.1109/COMST.2019.2896380
15. Li T, Wu B, Yang Y, Fan Y, Zhang Y, Liu W (2019) Compressing convolutional neural networks via factorized convolutional filters. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp 3977–3986.
16. 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
17. Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53(8):5455–5516
18. Scherer D, Müller A, Behnke S (2010) Evaluation of pooling operations in convolutional architectures for object recognition. International conference on artificial neural networks. Springer, pp 92–101
19. Ng A (2011) Sparse autoencoder. CS294A Lecture notes 72: 1–19
20. He D et al (2019) Intrusion detection based on stacked autoencoder for connected healthcare systems. IEEE Netw 33(6):64–69. https://doi.org/10.1109/MNET.001.1900105
21. Preethi D, Khare N (2020) Sparse auto encoder driven support vector regression based deep learning model for predicting network intrusions. Peer-to-Peer Netw Appl 1–11
22. Van NT, Thinh TN (2020) Temporal features learning using autoencoder for anomaly detection in network traffic. International conference on green technology and sustainable development. Springer, pp 15–26
23. Ramamurthy M, Robinson YH, Vimal S, Suresh A (2020) Auto encoder based dimensionality reduction and classification using convolutional neural networks for hyperspectral images. Microprocess Microsyst 79:103280
24. Lee SJ et al (2020) IMPACT: Impersonation attack detection via edge computing using deep autoencoder and feature abstraction. IEEE Access 8:65520–65529
25. Xu C, Shen J, Du X (2020) A method of few-shot network intrusion detection based on meta-learning framework. IEEE Trans Inf Forensics Secur 15:3540–3552. https://doi.org/10.1109/TIFS.2020.2991876
26. Zavrak S, İskefiyeli M (2020) Anomaly-based intrusion detection from network flow features using variational autoencoder. IEEE Access 8:108346–108358
27. Dutta V, Choraś M, Kozik R, Pawlicki M (2020) Hybrid model for improving the classification effectiveness of network intrusion detection. In: Cham ÁH, Cambra C, Urda D, Sedano J, Quintián H, Corchado E (eds) 13th international conference on computational intelligence in security for information systems (CISIS 2020), 2021//2021: Springer International Publishing, pp. 405–414.
28. Khan FA, Gumaei A, Derhab A, Hussain A (2019) A novel two-stage deep learning model for efficient network intrusion detection. IEEE Access 7:30373–30385. https://doi.org/10.1109/ACCESS.2019.2899721
29. Yang Y, Zheng K, Wu B, Yang Y, Wang X (2020) Network intrusion detection based on supervised adversarial variational auto-encoder with regularization. IEEE Access 8:42169–42184. https://doi.org/10.1109/ACCESS.2020.2977007
30. Li X, Chen W, Zhang Q, Wu L (2020) Building auto-encoder intrusion detection system based on random forest feature selection. Comput Sec 101851
31. Dong Gongl LL, Vuong L, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, Anton van den Hengel (2019) Memorizing Normality to Detect Anomaly: Memory-augmented Deep Autoencoder (MemAE) for Unsupervised Anomaly Detection,“ presented at the IEEE/CVF International Conference on Computer Vision (ICCV), 2019.
32. 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), 3–5, pp 1–5, doi: https://doi.org/10.1109/ICASERT.2019.8934495.
33. Alkadi O, Moustafa N, Turnbull B, Choo KR (2020) A deep blockchain framework-enabled collaborative intrusion detection for protecting IoT and cloud networks. IEEE Int Things J, pp 1–1. doi: https://doi.org/10.1109/JIOT.2020.2996590.
34. Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Topics Comput Intell 2(1):41–50. https://doi.org/10.1109/TETCI.2017.2772792
35. Yao H, Fu D, Zhang P, Li M, Liu Y (2019) MSML: a novel multilevel semi-supervised machine learning framework for intrusion detection system. IEEE Int Things J 6(2):1949–1959. https://doi.org/10.1109/JIOT.2018.2873125
36. Moustafa N, Turnbull B, Choo KR (2019) An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Int Things J 6(3):4815–4830. https://doi.org/10.1109/JIOT.2018.2871719
37. Muhammad G, Hossain MS, Garg S (2020) Stacked autoencoder-based intrusion detection system to combat financial fraudulent. IEEE Int Things J 1–1 doi: https://doi.org/10.1109/JIOT.2020.3041184.
38. Singh P, Kaur A, Aujla GS, Batth RS, Kanhere S (2020) DaaS: dew computing as a service for intelligent intrusion detection in

edge-of-things ecosystem. IEEE Int Things J: 1–1. doi: https://doi.org/10.1109/JIOT.2020.3029248.
39. Yu F, Koltun V (2015) Multi-scale context aggregation by dilated convolutions. *arXiv preprint* 1511.07122
40. Vaswani A et al. (2017) Attention is all you need. In: Presented at the 31st international conference on neural information processing systems. http://arxiv.org/abs/1706.03762.
41. Fu J et al. (2019) Dual attention network for scene segmentation. In 2019 IEEE/CVF conference on computer vision and pattern recognition (CVPR), pp 15–20 2019, pp 3141–3149. doi: https://doi.org/10.1109/CVPR.2019.00326.
42. Sun Y, Mao H, Guo Q, Yi Z (2016) Learning a good representation with unsymmetrical auto-encoder. Neural Comput Appl 27(5):1361–1367. https://doi.org/10.1007/s00521-015-1939-3
43. Injadat M, Moubayed A, Nassif AB, Shami A (2020) Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans Netw Serv Manag 1–1. doi: https://doi.org/10.1109/TNSM.2020.3014929.
44. 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) 27 Oct.–2 Nov. 2019, pp 1705–1714. doi: https://doi.org/10.1109/ICCV.2019.00179.
45. NDAE source code: https://github.com/ngoctn-lqdtu/A-Deep-Learning-Approach-to-Network-Intrusion-Detection
46. MemAE Source code: https://github.com/donggong1/memae-anomaly-detection
47. 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), 10–12 Nov 2015, pp 1–6. doi: https://doi.org/10.1109/MilCIS.2015.7348942.