Iranian Journal of Science and Technology, Transactions of Electrical Engineering (2022) 46:829–846
https://doi.org/10.1007/s40998-022-00498-1

RESEARCH PAPER

A Deep Learning Approach to Network Intrusion Detection Using a Proposed Supervised Sparse Auto-encoder and SVM

Ali Ghorbani[1] · Seyed Mostafa Fakhrahmad[1]

Received: 3 August 2020 / Accepted: 14 March 2022 / Published online: 20 May 2022

Abstract
Due to the increasing use of communication technologies for data transmission, security threats have increased over the past decade. One of the essential solutions to detect threats is NIDSs. Over the past few years, much research considered unsupervised feature extraction for NIDS like sparse auto-encoders; however, there is no research on the supervised auto-encoder methods. In this work, we propose a novel supervised sparse auto-encoder, which aims to extract more useful information for classification models than unsupervised methods. The proposed approach validated using *NSL-KDD, KDDCUP'99, CICIDS2017* data-sets, in the term of detection rate, and also response time. Experimental results in detection rate and test time are promising. In the case of binary classification, the accuracy of 90. 11% on *NSL-KDDTest*$^{+}$ and 91.21 on CICIDS2017 were achieved, which is a drastic improvement compared to state of the art with the more complex models and unsupervised representation learning models. Also, the result on 5class classification was considered.

Keywords Network intrusion detection · Deep learning · Sparse auto-encoder · Supervised auto-encoder

1 Introduction

Since 1994, the internet has been used to present different services to millions of users. Especially in the past decade, it changes our lives forever, from social media to how we work, buy our needs, and online banking. The dark side of this fact is the threats that target our data and privacy, which implements the need for network security systems. Any of these systems includes three parts, which are firewalls, antivirus software, and network intrusion detection systems (NIDSs) (Xin et al. 2018).

A NIDS, which plays an essential role in network security systems, has three primary functions: monitoring, analysis, and response (Alom et al. 2015). In this research, we consider the Analysis function of a NIDS. According to the way NIDS analyzes the data, there are three types of NIDSs.

1. Signature-based NIDS (SNIDS): comparing the received network traffic with an updated database of attack rules (Ahmed et al. 2016). Ex. Snort (Snort—Network Intrusion Detection & Prevention System 2022). Low false positive and high response time are advantages, and the crucial need for updates and unavailability to detecting zero-day attacks are disadvantages of this kind of NIDSs.
2. Anomaly detection based NIDS (ADNIDS): the kind of NIDSs trying to learn benign behaviors pattern, and any network traffic has deviated from this pattern detected as an attack (Hawkins et al. 2002; Moradi and Zulkernine 2004). It's common for learning this pattern to use a machine learning model or combing of models (hybrid model) (Aljawarneh et al. 2018). These models would adapt to the changing nature of network attacks (Leung and Leckie 2005). However, the need for high detection accuracy, low false-positive rate, and fast response are the challenges of an ADNIDS.
3. Hybrid-based NIDS (HNIDS): Combing SNIDS and ADNIDS provide an HNIDS (Viegas et al. 2016).

✉ Seyed Mostafa Fakhrahmad
fakhrahmad@shirazu.ac.ir

Ali Ghorbani
a.ghorbani@shirazu.ac.ir; Alighorbani29@gmail.com

[1] Department of Computer Science and Engineering and IT, School of Electrical and Computer Engineering, Shiraz University, Shiraz, Iran

Many shallow and deep learning models like SVM (support vector machine) (Mohammed and Sulaiman 2012; Wang et al. 2017), DTS (decision trees) (Eesa et al. 2015; Sindhu et al. 2012), Naïve Bayes (Louvieris et al. 2013; Xiao and Chen 2014; Yin et al. 2017), fuzzy logic (Ali et al. 2014; Elhag et al. 2015), KNN (K-nearest neighbors) (Aburomman and Reaz 2016; Li et al. 2014), CNN (convolutional neural network) (Li et al. 2017; Potluri et al. 2018; Vinayakumar et al. 2017; Wu et al. 2018; Zhu et al. 2018), DBN (Deep belief net) (Moradi and Zulkernine 2004), RNN (recurrent neural network) (Tang et al. 2018; Yin et al. 2017), non-symmetric deep auto-encoder (Shone et al. 2018) have been used for designing ADNIDSs.

One of the main features of many previous works on ADNIDSs was designing a powerful feature selector or extractor because of their effect on improving both accuracy and response time by finding a good representation of features and reducing the misleading data to help classifier (Tsai et al. 2009). Different kinds of feature learning techniques have been used for finding the best representation of data (Coates et al. 2011), which could be based on deep learning models, like restricted Boltzmann machine (Smolensky 1986), variational auto-encoders (Kingma and Welling 2013), convolutional neural networks (Goodfellow et al. 2016), sparse auto-encoders (Ng 2011) or not. Any feature learning algorithms can be categorized into two groups, which are linear and non-linear. The linear algorithms try to optimize $z = wx + b$ where z is our latent space and x is the original data. The problem with linear algorithms is that the optimal latent space must be in the space spanned by the input and when it is not the case, it provides too large error (Ng 2011). In non-linear groups, the problem is formulated as Eq. 1.

$$z = w\phi(x) + b, \tag{1}$$

wherein ϕ could be any non-linear functions (most popular is sigmoid), unlike the linear groups, the span is not limited to the span of the inputs any more (Yu and Principe 2019).

Another way to categorize feature learning is to use labels, which categorized feature learning into supervised, unsupervised, and semi-supervised methods. The supervised methods are beneficial when z is using for a supervised task like classification because, unlike the unsupervised techniques, it's also considered labels for extract features. To the best of our knowledge, there is only one paper that used semi-supervised feature learning (Nadeem et al. 2016), and there is no paper on supervised future learning on NIDSs till now, which is the primary motivation of our work.

The contributions of our paper are as follow:

i. A novel supervised sparse auto-encoder weighs the loss term with each feature's mutual information with the label.
ii. Combine this model with SVM (shallow and deep learning models).
iii. Use our model as an ADNIDS using NSL-KDD, KDDCUP' 99, and CICIDS2017 data sets.
iv. Design another ADNIDS using previously supervised auto-encoders to be a fair comparison with our model.
v. Compare our model with (iv), and a few of previous approaches and effect of our supervised sparse auto-encoder on the number of support vectors compare to SVM and simple, sparse auto-encoder SVM; also the comparison on zero-day attacks detection rate of the proposed model with these two approaches has been provided.

The rest of the paper is structured as follows. Section 2, reviews previous related works. In Sect. 3 we discussed background information. Section 4 specifies our novel proposed solution; Sect. 5 shows the proposed model's evaluation and result. The comparison discussed is provided in this section. Finally, Sect. 6 concludes the proposed model and discussed plans for improving the proposed model.

2 Related Work

As mentioned before, the NIDSs are the most essential and challenging part of network security systems, making them a fascinating topic for researchers. Especially in the area of designing ADNIDSs, which need to use machine learning models. Various approaches are used to creating ADNIDSs, like different shallow learning models and deep learning models. The shallow models are faster than deep learning models, but they suffered from a low detection rate. In this section, we will discuss some of the notable works in this area.

Tavallaee et al. (2009) provided a statistical analysis of the *KDDCUP' 99* data set. They found two critical issues on this data set that affect evaluated systems' performance on real-world data: a vast number of redundant data biassed towards the more frequent records and prevents it from learning unfrequented records. The second problem is the test set and the train set are not independent of each other, which means the test and train set source is similar. These problems cause high detection rates in research that used this data set and low detection rate in practice. To solve these issues, they proposed *NSL-KDD* data set. They also tested this data set's performance on multiple shallow learning models like j48, naïve Bayes, NB tree.

Lin et al. (2015) provided NIDS based on combining cluster centers and k-nearest neighbor. After using a clustering technique to extract cluster centers (the number of clusters is equal to the number of labels), they replaced all features in the data set with only one feature. This feature is the sum of distances of each element with its cluster center and its nearest neighbor. They used KNN for the classification task. The detection rate and response time were noticeable if considering clustering as preprocessing. They evaluated their model on *KDDCUP' 99*. Another example of using KNN is Liao and Vemuri (2002).

Alom et al. (2015) proposed a model based on the deep belief net (DBN). They compared their results with SVM and DBN-SVM. They reached 97.5% accuracy when using 40% of *NSL-KDDTrain*$^+$ data as train set and the rest as the test set, which is hurt the main idea behind the *NSL-KDD* data set (make train and test set independent from each other). Alrawashdeh and Purdy (2016) provided a model based on the restricted Boltzmann machine(RBM) and logistic regression (LR). The evaluated data set on this research was *KDDCUP' 99*.

Kim et al. (2016) provided a model using the long-short memory term (LSTM) architecture on a recurrent neural network (RNN). The *KDDCUP' 99* data set was used to evaluate their model. They compared their approach with the general regression neural network (GRNN), probabilistic neural network (PNN), KNN, SVM, etc.

Yin et al. (2017) Also used RNN with softmax regression for developing a NIDS. They evaluated their model on the NSL-KDD data set. The comparison between their model and some of the shallow learning models like j48, naïve Bayes, NB tree, random forest, and the random tree is also provided. The improvement of accuracy was noticeable, but the model was suffered from very high response time, which comes from the nature of RNN.

Wu et al. (2018) proposed a CNN-based model evaluated on the *NSL-KDD* data set. They turned each Data set sample, with 123 features to an 11×11 image by removing one feature with the lowest variance coefficient. They proposed a model with two convolutional and two polling layers and one fully connected layer for the classification task. They also consider weights for samples in cost function according to the proportion of different class samples to solve the imbalanced data set problem. Li et al. (2017) also used CNN to design a NIDS. They also used the *NSL-KDD* data set. They turned each sample of the data set to a vector of binaries after desterilizing continuous features and hot encoding of non-numerical features. They used two famous CNN based models (ResNET, GoogleNET) to classify data.

Shone et al. (2018) proposed a novel auto-encoder named non-symmetric deep auto-encoder. This auto-encoder encoding layers is not equal to the number of decoding layers to design a NIDS. They used *NSL-KDDTrain*$^+$ as train set and *NSL-KDD* test with 18,794 instances as test data set, which is the data set that removed the attack types that consist in *NSL-KDDTest*$^+$ but not in NSL-*KDDTrain*$^+$ (zero-day attacks). The result was noticeable compared to the deep belief net.

Yousefi-Azar et al. (2017) proposed a feature learning model based on deep auto-encoders. They used multiple shallow learning models for the supervised task like Gaussian naïve Bayes and SVM.

Farahnakian and Heikkonen (2018) provided a model based on deep auto-encoders. They used four stacked auto-encoder for pre-training. After pre-training, they used the soft-max layer for classification and also fine-tuned all layers. The result of their model was promising on KDDCUP' 99, but it was expensive in the term of computation compare to our model.

Aygun and Yavuz (2017) used auto-encoder and denoising auto-encoder, respectively, to propose a NIDS. They used an auto-encoder to learn normal behavior. After that, they find a threshold for reconstruction error. In test time, they used this threshold to classify data. The NSL-KDD data set was used as a data set of this research.

Nadeem et al. (2016) proposed a model based on the ladder network (Rasmus et al. 2015; Valpola 2015), a semi-supervised model. The model was evaluated on *10% KDDCUP' 99* data set. Their model's main issue was suffering from high response time, as was mentioned in their paper.

Al-Qatf et al. (2018) provided a model using sparse auto-encoder and SVM using *NSL-KDD* data set. Their model has less accuracy and converges slower in pre-training compared to our model. Javaid et al. (2016) also used the sparse auto-encoder combined with the softmax layer. They also used the *NSL-KDD* data set.

Tang et al. (2020) proposed a LightGBM auto-encoder-based model for NIDS. They used the LightGBM model for feature selection and the AE model for training and detecting. They used a threshold on AE reconstruction error for distinguishing between normal and attack behaviors. For evaluation, the Nsl-KDD data set has been used. The architecture of their AE model consists of 5 hidden layers. They achieved an 89.82% accuracy rate, which is considerable. However, as the LightGBM models are computationally complex (these models are based on gradient boosting algorithms), their AE model contains five layers. The Realtimeness of NIDSs is not considered in this research. Also, their model is disabled to face multiclass classification problems.

Khraisat et al. (2020) proposed a model by combining the c5 decision tree and one-class SVM. They evaluate their model using the NSL-KDD data set, with an 83.24% accuracy rate on binary classification.

Ieracitano et al. (2020) proposed a model based on statistical analysis for feature extraction from 122 to 102 dimensions and AE for dimensional reduction from 102 to 50. The NSL-KDD was used as the data set in this research. The accuracy rate was 84.21% in binary classification.

Li et al. (2020) design a model using the NSL-KDD data-set using a multi convolutional Neural Network (multi-CNN). They achieved an 89.95% accuracy rate for binary classification and 81.32% for 5-class classification. Obviously, the response time was not considered in this method because one CNN is very complex; and multiple of them have been used.

Zhou et al. (2020) proposed a method based on M-Adaboost, which is a complex ensemble model. They used NSL-KDD Train^{+} for both training and testing.

We understood after studying these approaches that many of these approaches suffering from low accuracy converge slowly in pertaining or high response time.

For having a comparison with some of the following approaches, we evaluated our model on *NSL-KDD* and *KDDCUP' 99* data sets in two approaches using tenfold cross-validation on training sets of these data sets and using training sets for training and test sets for testing. We examined both binary and five category classification problems.

3 Background

This section provides information about the proposed method's base concepts, which is necessary for understanding the motivation and concepts behind our work. First, a brief elucidation about deep learning and multiple representation learning methods is based on it; After that, soft-margin SVM will be discussed. Finally, the mutual information will be discussed.

3.1 Deep Learning

Deep learning is a subcategory of machine learning that tries to find high-level features from input data (Deng and Yu 2014). Deep learning is based on neural networks (stochastic models that inspire directly from human and animal neural systems intending to find a relationship between input and output). Any neural network has two paths, forward and backward. The input has been mapped to output by crossing over a graph with multilevel linear or non-linear transformation layers in the forwarding path. The backward path calculates the gradient of the cost for the weights and tries to minimize the cost by updating weights. In other words, it is based on feature and representation learning in different layers of the model. Deep learning models divide each complex concept into more straightforward concepts. This process leads the model to find essential concepts. In recent years deep learning achieved considerable success in many fields like machine vision, machine translation, and speech recognition because of the availability to find intricate data patterns. We used the power of deep learning in representation learning. In the following, we discussed some of the previous methods in representation learning based on deep learning.

3.1.1 Auto-encoder

Auto-encoders are unsupervised models that try to learn a good representation of data by mapping the data from the original space to a latent space (encoding) and reconstruct the original data from latent space (decoding) (Baldi 2012; Hinton and Zemel 1997) as it is shown in Fig. 1. Any auto-encoder could have three or more layers, an input layer, an output layer, hidden layer(s). If only linear functions or one sigmoid function used in the middle layers, the auto-encoder acts like PCA (principal component analyzer). It was considering F as encoding function with w and b parameters for mapping data to latent space according to Eq. 2.

$$z = F_{w,b}(x). \tag{2}$$

Also g as decoding function with $w^{'}$ and $b^{'}$ parameters try to reconstruct data from latent space according to Eq. 3.

$$x^{'} = g_{w^{'},b^{'}}(z). \tag{3}$$

The loss function of the auto-encoder can be something to minimize the difference between x and $x^{'}$ like, mean square error. Auto-encoder's family can be used for feature extraction, data de-noising, and a generative data model.

3.1.2 Sparse Auto-encode

Sparse auto-encoders are auto-encoders trying to discover exciting structures by imposing other constraints on the network. For example, when the latent space dimensions are higher than the original space, the vanilla auto-encoder desired to copy input in latent space and adding some zeroes to latent space. This problem can be handled by adding a sparsity constraint on the hidden layer. This sparsity is a Kullback–Leibler divergence between the average of activated nodes in the hidden layer and a fixed

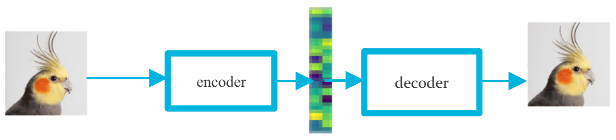

Fig. 1 example of an auto-encoder

number between 0 and 1. So the loss function of it could be like Eq. 4.

$$J = \frac{1}{2m}\sum_{i=1}^{m}\|x_i - \widehat{x}_i\|^2 + \beta\sum_{j=1}^{k} kl(\rho\|\widehat{\rho}_j), \tag{4}$$

β is the constants considering as hyper-parameters of the loss function that show the importance of respective terms. The first term is a mean square error between original and reconstructed data, and the second term is the sparsity constraint will calculate according to Eq. 5.

$$kl(\rho\|\widehat{\rho}_j) = \rho\log\left(\frac{\rho}{\widehat{\rho}_j}\right) + (1-p)\log\left(\frac{1-\rho}{1-\widehat{\rho}_j}\right), \tag{5}$$

where ρ is sparsity constraint parameter and $\widehat{\rho}_j$ is the average value of activated hidden unit j overall training input x (Ng 2011).

3.1.3 Supervised Auto-encoder

In this kind of auto-encoders, there is a supervised constraint in the loss function. This kind of auto-encoders is mostly used when the latent variable is used as supervised model input. Ex (Le and Patterson 2018) Adding a softmax layer to the network connected to the hidden layer and reformulates the loss as Eq. 6.

$$J = \frac{1}{2m}\sum_{i=1}^{m}\|x - \widehat{x}_i\|^2 + \frac{\beta}{2m}\sum_{i=1}^{m}\|y - \widehat{y}_i\|^2. \tag{6}$$

3.2 SVM (Support Vector Machine)

A classifier aimed to find the best separating hyperplane that maximizes the margin between two classes. Figure 2 shows a schema of how this classifier works. W and b are considered as parameters of the hyperplane that need to be learned. After learning, any new sample could be classified into each positive or negative class by Eq. 7.

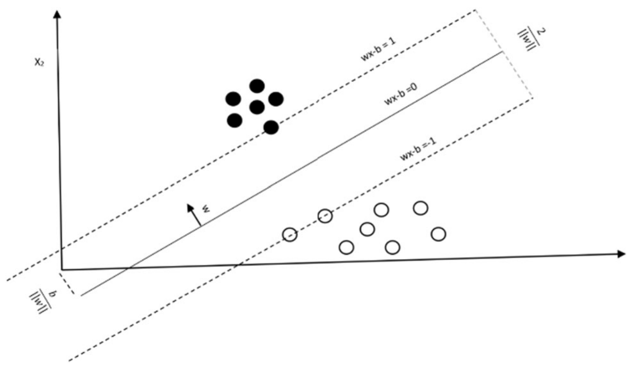

Fig. 2 A geometric view of a SVM classifier

$$\forall i y^{(i)}\left(w^T x^{(\mathrm{i})} + b\right) \geq 1 \tag{7}$$

SVM aims to satisfy two requirements:

i. Maximize the distance between decision boundaries, which is $\frac{2}{\|w\|}$.
ii. Correctly classify all $x^{(i)}$.

To achieve these goals, SVM formulated a problem that needs to be solved using the primal form shown in Eq. 8.

$$\begin{aligned} & min_{w.b}\frac{\|w\|}{2} + C\sum_{i=1}^{N}\varepsilon^{(i)} \\ & s.t y^{(i)}\left(w^T\varphi(x^{(i)}) + b\right) \geq 1 - \varepsilon^{(i)} \\ & \varepsilon^{(i)} > 0 \\ & \forall i \in \{1,\cdots,N\} \end{aligned} \tag{8}$$

where $\varepsilon^{(i)}$ is slack variables and C is error penalty term, and φ is kernel function that maps data in the space can be separated better. For solving this problem, first, it needs to be written in the dual form, which is considered in Eq. 9.

$$max_{\alpha}min_{w.b}\frac{\|w\|}{2} + C\sum_{i=1}^{N}\alpha^{(i)}\left(1 - \left(w^T\varphi\left(x^{(\mathrm{i})}\right) + b\right)\right). \tag{9}$$

After solving this problem, the optimal values for w looks like Eq. 10.

$$w_i = \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\varphi\left(x^{(i)}\right). \tag{10}$$

Also, in the test time, the prediction of each label is calculated by Eq. 11.

$$y_{test} = \sum_{i=1}^{N}\alpha^{(i)}y^{(i)}\varphi\left(x^{(i)}\right)^T\varphi(x_{test}) + b, \tag{11}$$

$\alpha^{(i)} > 0$ if and only if $x^{(i)}$ Is a support vector (Chapelle 2007). So less number of support vectors concludes fewer computations in the test.

3.3 Mutual Information

The mutual information between two random variables is a measure of how much information can be gotten from one of them by observing another one. It is calculated by Eq. 12 (Cover and Thomas 2012). High mutual information indicates a massive reduction in uncertainty; low mutual information indicates a small reduction, and zero means independence.

$$I(x;y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log(\frac{p(x,y)}{p(x)p(y)}). \quad (12)$$

4 Proposed Methodology

This section will describe our supervised sparse auto-encoder and its benefits in two parts; Sect. 4.1 describes our proposed supervised sparse auto-encoder and the motivation of the proposed methodology, and Sect. 4.2 provides the benefits of the proposed model compared to the traditional supervised auto-encoders.

4.1 Sparse Auto-encoder with Mutual Information Weight Reconstruction Loss

The main challenges ahead of designing NIDSs are reliability and being real-time. So we aim to propose a method that could meet the expectations of both of the challenges. This paper introduces our supervised sparse auto-encoder, an auto-encoder with a sparsity constraint, and weighted reconstruction error with the diagonal matrix that shows each feature's importance in the classification tasks. One of the reasons that motivate us to propose our approach was when an unsupervised auto-encoder tries to find an optimal solution, it has no idea about which features are more important than the others in the supervised (classification) task, and it must be more information about them in the latent variable. These models only try to minimize loss, and there is a possibility to be no information about a critical feature in the latent variable. This disadvantage of unsupervised auto-encoder significantly affects the models designed for NIDSs. There is no need to use unsupervised representation learning in the training process when the labels are available. Representation learning is mostly used along with a supervised classification model. So our approach is going to put a positive impact on supervised tasks by having more critical information about the label than the unsupervised auto-encoder. This process can also reduce classifier complexity by giving it more profitable information, which causes finding a solution easier (in our case number of support vectors shows that).

Another advantage of our model is that the weight matrix can be any matrix that shows the importance of different features. We test joint entropy, correlation, and mutual information of features with the label. As we thought, mutual information has the most effect on accuracy and number of support vectors.

The weight matrix will be calculated as a preprocessing task, which is a diagonal matrix that elements i, i shows the mutual information of ith feature with the label. Considering f_i as ith feature the matrix is going to be calculated as follows;

$$M_{i,j} = \begin{cases} I(f_i; y) & \text{if } f_i = j \\ 0 & \text{if } f_i \neq j \end{cases}. \quad (13)$$

Considering matrix as M, the loss function of the proposed model formulated as follows;

$$J = \frac{1}{2m}\sum_{i=1}^{m}(\mathrm{x_i} - \widehat{\mathrm{x}}_\mathrm{i})^\mathrm{T} \times \mathrm{M} \times (\mathrm{x_i} - \widehat{\mathrm{x}}_\mathrm{i}) + \frac{\lambda}{2}(\sum_{kn} w^2 + \sum_{nk} v^2 + \sum_{k} b^2 + \sum_{n} b'^2) + \beta\sum_{j=1}^{k} kl(\rho \| \widehat{\rho}_j), \quad (14)$$

W and V are the weight matrix of the first and second layer, b and b' are also the bias vector of them, respectively.

When the auto-encoder begin the learning process, it gives more importance to the feature with more mutual information or other metrics; we decide for showing feature importance because these features have an immense weight than the others in the loss function, so for minimizing the loss the reconstruction of them is more important than the others. The auto-encoder needs to have more information about them in latent space to reduce the reconstruction error of corresponding features because the decoder uses the latent space to reconstruct data. This process may increase the risk of overfitting. We handle this risk by adding the regularizer to them. After the auto-encoder is well trained, the latent variable of training data will be used for training the SVM. In the test phase, each testing data first feeds the auto-encoders encoder to find the latent variable. The latent variable of corresponding testing data will feed to the SVM to predict the label. The idea is so simple, but our results show that our model finds the optimal latent space faster than the others, reduces classifiers complexity, and is better or at least competitive with other works even with more hidden layers. The Alghorithm 1 shows the pseudocode of the proposed method.

```
Alghortihm1. Proposed Supervised SPAE for NIDS

Pre_Trainging: input: x_train, x_test, y_train;
output: M, X_train, X_test

X_train = pre_processing(x_train )
X_test = pre_processing(x_test)
$M_{i,j} = \begin{cases} I(f_i; y\_train) & if\ i = j \\ 0 & if\ i \neq j \end{cases}$
Return M, X_train, X_test

Training: input: X_train, M, y_train;
 Output: SSPAE, SVM_Model

SSPAE ← SSPAE.Train(X_train, M)
Z_train ←SSPAE.Transform(X_train)
SVM_Model = SVM.Train(Z_train, y_train)
Return SVM_Model, SSPAE

Testing:
 Input x_test, SVM_Model, SSPAE;
output: y_predict

Z_Test = SSPAE.Transform(X_test)
y_predict = SVM_Model.predict(Z_test)
Return y_predict
```

4.2 Benefits of the Proposed Model Compared to Traditional Supervised Auto-encoders

As was discussed before, in previous models, a supervised constraint was added to the auto-encoder like Eq. 6, so the pre-training and training phase had been merged. These models have a high potential to over-fit on training data because of the label's direct effect on encoder layers. However, in our models, the labels will not directly affect the encoder and decoder layers but only tell which features it preferred better than the others.

Another benefit of our model over the traditional ones is that in traditional supervised auto-encoders, the supervised term acts as a constant in decoder layers, which causes no effect on backpropagation in these layers. However, in our model, it affects both decoder and encoder layers, which causes less risk on overfitting of the encoder and make the decoder a better generative model (which is not the consideration on NIDSs) as a consequence.

5 Evaluation and Result

Our proposed model, implemented using GPU-enabled TensorFlow is performed on 64-bit Ubuntu 14.04 on PC with Intel(R) Core(TM) i5-6400 CPU 2.70 GHz, 16 GB ram, and NVIDIA GTX 1070 GPU. The hyper-parameters for classification and supervised sparse auto-encoder (SSAE) have been provided in Table 1. Also, RMS-probe with exponential decay learning rate (initial 0.001 for binary and 0.01 for 5 class classification) is used as an auto-encoder optimizer. All of these hyper-parameters are found by grid search of different values using validation data.

This section provides details of our implementation and also the result and comparison with other researches. Section 5.1, the evaluation metrics were discussed, Sect. 5.2 gives the detail of data sets, Sect. 5.3 discussed preprocessing that needs to be done to prepare the data. Finally, the result and comparison were provided in Sect. 5.4.

5.1 Evaluation Metrics

To perform our evaluation, we used *NSL-KDD* and *KDDCUP' 99* data-sets. Our approach's performance was evaluated using the following metrics, which have been used in other research in the same field.

1. True Positive (TP): anomaly records that are correctly classified as an anomaly.
2. False Positive (FP): normal records that are incorrectly classified.
3. True Negative (TN): normal records that are correctly classified as normal.
4. False Negative (FN): anomaly records that are classified incorrectly.

We used these metrics to compute the following measures:

1. Accuracy rate (AC): Percentage of correctly classified records overall records Formula (Metz 1978).

$$\mathrm{AC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \times 100\% \tag{15}$$

2. Precision rate (*P*): Percentage of correctly classified anomaly records over a total of classified anomalies (Powers 2020).

$$P = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \times 100\% \tag{16}$$

3. Recall (*R*): Percentage of correctly classified anomalies divided by the number of attack entries (Powers 2020).

Table 1 hyper-parameters of the proposed model

	NSL-KDD		KDDCUP99	CICIDS2017
	Binary	5-class	Binary	Binary
SSAE				
AE-arch	122–30-122		122–30–122	79–100–50–25–50–100–79
λ	6×10^{-4}		6×10^{-4}	6×10^{-4}
β	3		0.3	0.3
$\widehat{\rho}_j$	0.5		0.77	0.77
SVM				
c	10	5.56	10	1
γ	4	1.06	4	0.1

$$R = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \times 100\% \tag{17}$$

4. F-measure (F): Harmonic mean of precision and recall, which is considered the most crucial measure for NIDSs (Powers 2020).

$$F = \frac{2 \times P \times R}{P + R} \times 100\% \tag{18}$$

5. False-positive rate:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{19}$$

6. False-negative rate:

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{TP} + \mathrm{FN}} \tag{20}$$

We used all of the following metrics for evaluation on NSL-KDD and CICIDS2017. However, because the goal of using KDDCUP99, which is the old version of NSL-KDD, was the comparison with related works, we only used the accuracy metric on it.

5.2 Data Sets

In this paper, *KDDCUP' 99, NSL-KDD*, and CICIDS2017 data-sets were considered for evaluation. These data sets have been used Widely in NIDSs research areas.

5.2.1 KDDCUP' 99

Since 1999 *KDDCUP' 99* has been the most used data set for the evaluation of NIDSs. This data set was evaluated by using data captured in DARPA's 98 programs (Stolfo et al. 2000). The Data consists of 4 gigabytes-worth of compressed tcpdump data resulting from 7 weeks of network traffic. The training data set involve 4,900,000 records each consider as a connection record with 41 feature includes Basic, Domain knowledge and time observation features. Each record was labeled as either Attack from 22 different attack types or normal behavior. It is common to use *10% KDDCUP' 99* data set with 145,586 records with no duplicated data and a great representation of the complete data set as training data set and *KDDCUP' 99* corrected with 77,291 records as testing data set. The features in this data set involve numerical and symbolic values, so it needs preprocessing for evaluating our model. The numerical values need to be normalized to reduce computational complexity. The whole process of preprocessing will be explaining.

5.2.2 NSL-KDD

The newer version of KDDCUP' 99 to overcome the KDDCUP' 99 problems (Tavallaee et al. 2009). This data set structure is similar to the previous data set, consisting of 41 features; three of these features are symbolic, and others are numeric. As discussed before, the need for preprocessing is crucial for this data set. NSL-KDD data set consists of two different training data set, which are KDDTrain^{+} and KDDTrain^{+}20%, and also two testing data sets, which are KDDTest^{+} and KDDTest^{-21}. In this research, we used KDDTrain^{+} as training data set and KDDTest^{+} as testing data set. The training data set consist of 22 different types of attacks, and the testing data set consist of 34 attack types, which means the KDDTest^{+} data set consist of 12 attacks that do not appear in the training data set (zero-day attacks), so the generalization power of model on zero-day attacks will be considered. These attacks will categories into four categories, which are r2l, u2r, Dos, and probe. The KDDTrain^{+} data set consist 125,973 records and KDDTest^{+} data set 22,544 records. The description of the following data sets is provided in Table 2.

Table 2 Description of NSL-KDD and 10%KDDCUP data sets

Category	10% KDDCUP		NSL-KDD		
	Train	Test	Train$^+$	Test$^+$	Test$^+$ no zero-day attacks
DoS	54,572	23,570	45,927	7458	5741
Probe	2131	2682	11,656	2421	1106
R2L	999	3056	995	2754	2199
U2R	52	70	52	200	37
Normal	87,832	47,913	67,343	9711	9711
Total	145,586	77,291	125,973	22,544	18,794

5.2.3 CIC IDS 2017 (Sharafaldin et al. 2018)

CIC IDS 2017 is the newer version of ISCX 2012 (Shiravi et al. 2012) provided by the Canadian Institute of Cybersecurity, consist of a variety of benign and malicious network traffic types. The attack types in this data set are the most up to date attack scenarios. The Data set includes eight CSV files; each file consists of 83 features and label, the label could be either benign or one of 14 different attack types also, 285,735 instances have a null label, so they have to be deleted from the data set. In this research, we first concatenate all files that provide us with a data set with 3,116,478 instances and 2,830,743 instances after deleting flows with the null label. Table 3 shows the details of this data set. As the Data is vast and cannot be used for training models, we randomly select 10% of the data instances but with the vision of selecting from different classes and distribution of original data set. Some of the features (flag id, destination IP, source IP, destination port, source port) of this data set needs to be deleted for designing NIDS because it is not correct to make the IDS model sensitive to them, for example, if the IDS is sensitive to the source IP the attacker can pass the IDS by changing his/her IP address. Similar to previous data sets, this data set also needs normalization.

Table 3 Description of CICIDS2017 data set

CSV file	#instances	#attacks	#type of attacks
Monday	529,918	0	All benign
Tuesday	445,909	7938	FTP-Patator
		5897	SSH-Patator
Wednesday	692,703	5796	DoS-slowloris
		5499	DoS-Slowhttptest
		231,073	DoS-Hulk
		10,293	DoS-GoldenEye
		11	Heartbleed
Thursday AM	170,366	1507	Web Attack-Brute Force
		652	Web Attack-XSS
		21	Web Attack-Sql Injection
Thursday PM	288,602	36	Infiltration
Friday AM	191,033	1966	Bot
Friday PM1	286,467	158,930	PortScan
Friday PM2	225,745	128,027	DDOS
Total	2,830,743	557,646	14 attack type

5.3 Data Preprocessing

5.3.1 Normalization

Many features of these data sets consist of a broad range between maximum and minimum, which achieves computational complexity during the learning and testing process. Therefore, we normalized these features using a min–max normalization map of each feature to range between 0 and one according to Eq. 21.

$$x_i = \frac{x_i - \min}{\max - \min} \tag{21}$$

where x_i is a data point, min is the minimum value from all data points, and max is the maximum value for each feature.

5.3.2 Categorize Labels into Five Categories

All the attacks in this data-sets fall in one of the following four categories(Tavallaee et al. 2009) according to Table 4:

1. Denial of Service Attack (DoS): is an attack that attackers target the system's availability by making resources of the victim too busy or full.
2. User to Root Attack (U2R): kind of attacks that attackers start the system with a regular user account and exploit some vulnerabilities to access root user privileges.
3. Remote to Local Attack (R2L): when the attacker can send packets to the system but do not have any accounts on the system, but gain against some vulnerabilities to local access as a user to the system
4. Probing Attack: kind of attacks that attacker investigate network over the system or the system to determine

Table 4 Category of attacks on NSLKDD

Category	Attacks
DoS	back, land, neptune, pod, smurf, teardrop, Mailbomb, Processtable, Udpstorm, Apache2, Worm
U2R	buffer-overflow, loadmodule, perl,rootkit, Sqlattack, Xterm, Ps
R2L	ftp-write, guess-passwd, imap, multihop, phf, spy, warezclient, warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendmail, Named
Probe	ipsweep, nmap, portsweep, satan, Portsweep, Mscan, Saint

vulnerabilities of the system that can lead to exploits, which could be circumventing system security controls.

5.3.3 One-Hot Coding Symbolic Labels

The NSL-KDD data set has 38 numerical features and three symbolic features to convert these symbolic features to numerical features; we used the one-hot coding strategy. EX. The "protocol-type" feature has three distinct attributes, these attributes be encoded as (1, 0, 0), (0, 1, 0), (0, 0, 1) binary vectors. This process converts NSL-KDD features set from 41 feature dimensional to 122-dimensional features.

5.4 Result and Comparison

5.4.1 Number of Support Vectors Training and Testing Time

As SVM with RBF kernel is used as the classifier, the number of support vectors (nSVs) and dimensions will be impacted training and testing time because SVM with non-linear kernels computation and usage memory grows relative to nSVs. So we compare our approach (SSPAE-SVM) in terms of nSVs, training, and testing time on *NSL-KDD* AN and CICIDS2017 data sets with simple SVM and sparse auto-encoder SVM (SPAE-SVM) (Al-Qatf et al. 2018) in four different approaches for NSL-KDD, and one approach for CICIDS2017. Also, we provide a comparison between the pre-training (representation learning) time of the proposed method and SPAE-SVM.

In the first approach, we used *NSL-KDDTrain*$^+$ as a training set and *NSL-KDDTest*$^+$ as a test set, also we consider CICIDS2017 10% 0.7 as training and 0.3 as testing data. The label is considered binary (normal and Attack). The result shows that we reduce the number of support vectors in all three models compared to simple SVM and SPAE-SVM in thereupon training and testing time. Also, the dimensional reduction using auto-encoder affects memory and computational complexity. Table 5 shows the result.

In the second approach, we used tenfold cross-validation on the *NSL-KDDTrain*$^+$, considering labels as binary like the previous approach; the result also emphasis on efficiency of the proposed model. Because pre-training time was similar to the first approach, these results were not considered again. Table 6 shows the result of this.

In the third approach, we used *NSL-KDDTrain*$^+$ and *NSL-KDDTest*$^+$ as the test set. The label is considering as five labels (normal and four attack categories). The one vs. rest strategy was used for the classification model. This result also shows the positive effect of our model on memory and time complexity. Table 7 shows the result. In the fourth approach, the *NSL-KDDTrain*$^+$ was used for both training and testing with a tenfold cross-validation strategy Table 8 shows the result. What is noticeable in all of these approaches is the improvement of our model in training and testing time of a NIDAS; e.g., the total pre-training and training time of our model is less than training time SVM.

5.4.2 Evaluation Considering Binary Classification

Two approaches were considered for binary classification, first using *NSL-KDDTrain*$^+$ and *NSL-KDDTest*$^+$ and CICIDS2017 10% data sets. In the second approach, the evaluation on *NSL-KDDTrain*$^+$ using tenfold-cross-validation strategy, in each fold 70% of data set used as training data and 30% as testing. Table 9 shows the result in the first approach as it shows the proposed model achieves 90.11% accuracy on NLSKDDTest$^+$ and 91.22% on CICIDS2017; which is very noticeable, considering only one-layer encoding for NSLKDD compare to other works using deep-stacked auto-encoders with more than one auto-encoder and more than one layers for encoding (Shone et al. 2018) or using multiple convolutional layers and overhead of turning samples to pictures (Wu et al. 2018). So the proposed model is powerful and also simple in terms of computation, which means that the proposed model improves both critical requirements of NIDSs. The

Table 5 Training and testing time and number of support vectors (NSV) comparison for SVM, SPAE-SVM, Proposed model (SSPAE-SVM) for binary classification on CICIDS2017, NSL-KDDTrain$^+$ and NSL-KDDTest$^+$

Data set (sec)	Method (nSVs)	Pre-training time (s)	Training time (s)	Testing time	nSVs
NSL-KDD	SVM	–	520.56	17.03	4596
NSL-KDD	SPAE-SVM	308.74	179.15	9.49	3879
NSL-KDD	SSPAE-SVM	268.36	182.08	5.22	3870
CICIDS2017	SVM	–	722.61	102.08	19,670
CICIDS2017	SPAE-SVM	421.32	541.71	70.82	18,022
CICIDS2017	SSPAE-SVM	291.22	317.36	46.54	16,550

Table 6 Training and testing time and number of support vectors (NSV) comparison for SVM, SPAE-SVM, proposed model for binary classification based on NSL-KDDTrain$^+$

Method	Training time (s)	Testing time (s)	nSVs
SVM	38,734	240.35	38,734
SPAE-SVM	1044.47	71.66	32,597
Proposed model	968.35	75	33,287

Table 7 Training and testing time and number of support vectors (nSV) comparison for SVM, SPAE-SVM, proposed model for 5 class classification based on NSL-KDDTest$^+$

Method	Pre-training time (s)	Training time (s)	Testing time (ss)	nSVs
SVM	–	2382.80	33.34	9049
SPAE-SVM	160.92	368.84	12.14	5957
Proposed model	120.65	290.42	6.33	5271

Table 8 Training and testing time and the number of support vectors (nSV) comparison for SVM, SPAE-SVM, proposed model 5 class classification based on NSL-KDDTrain$^+$

Method	Training time (s)	Testing time(s)	nSV
SVM	11,512.23	481.26	77,513
SPAE-SVM	767.15	97.48	48,339
Proposed model	730.239	93.087	46,234

comparison also shows the improvement of our model compare to SVM and SPAE-SVM.

The false-positive rate and false-negative rate of each model for *NSL-KDDTest*$^+$ and CICIDS2017 been shown in Figs. 3 and 4, respectively.

The ROC curve and AUC of the model on CICIDS2017 have also shown in Fig. 5, and ROC curve of *NSL-KDDTest*$^+$ has been shown in Fig. 6. ROC is a probability curve, and AUC represents the degree or measure of separability. It tells how much the model is capable of distinguishing between classes (Fig. 7).

Table 10 showed the result when *NSL-KDDTrain*$^+$ tenfold cross-validation was used. As the result shows we do not have such massive improvement as the previous approach, which is a good thing because it shows less potential for the over-fitting of our model.

5.4.3 Evaluation Considering 5 Class Classification

This approach Compared to our result in 5 class in terms of accuracy, precision, recall, and *F*-measure, considering these measures for different classes with SVM and SPAE-SVM. All of the results were provided considering different class labels. This evaluation also shows our improvement in 5 class classification when our model got 80.46% accuracy, the SVM got 77.49% accuracy, and SPAE-SVM got 78.54%, as has been provided in Fig. 8. Also, Fig. 9 compares the result in tenfold cross-validation on NSL-KDDTrain$^+$. ROC curve of the proposed model considering 5 class classification problems also has been provided in Fig. 5.

Table 9 Evaluation result considering binary classification on NSL-KDDTest$^+$ and CICIDS2017 data sets

Data set	method	Accuracy	Precision	Recall	*F*-measure
NSL-KDD	SVM	88.91	93.80	86.22	89.85
	SPAE-SVM	87.85	93.56	84.46	88.78
	SSPAE-SVM	90.11	93.74	88.53	91.06
CICIDS 2017	SVM	83.36	1	83.34	90.92
	SPAE-SVM	88. 19	93.30	92.58	92.94
	SSPAE-SVM	91.22	96.04	93.59	94.80

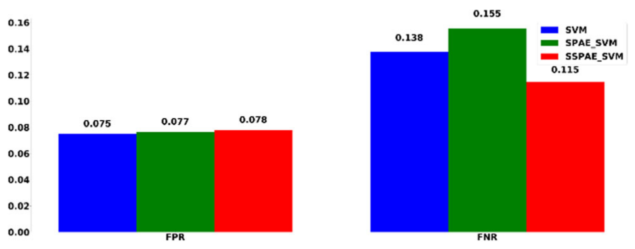

Fig. 3 FNR and FPR on binary classification in CICIDS2017

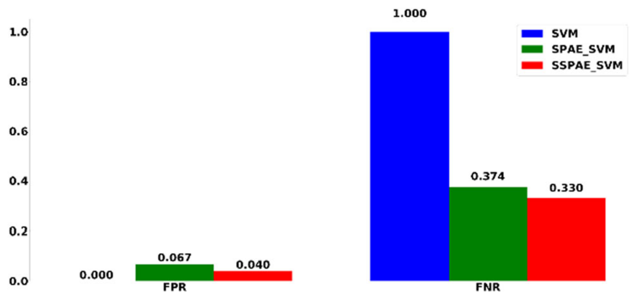

Fig. 4 FPR and FNR on binary classification in NSL-KDDTest^{+}

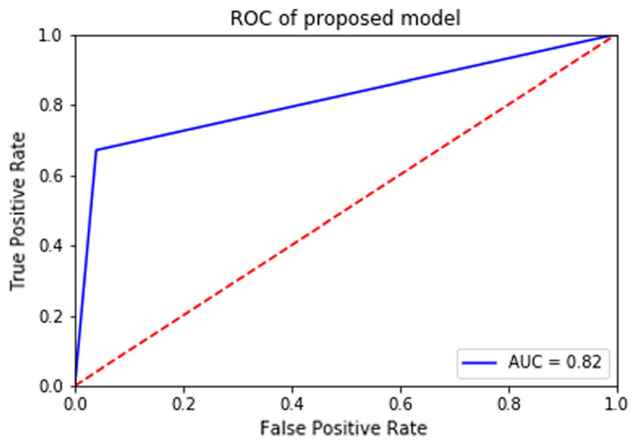

Fig. 5 ROC curve and AUC value of proposed model on binary classification based on CICIDS2017

5.4.4 The Zero-Day Attacks Detection Rate

This section compared the zero-day attack detection rate of our model to SVM and SPAE-SVM to show the proposed model's generalization power. From 22,544 instances of *NSL KDDTest*$^{+}$ 3750 are zero-day attacks, which means the attack types which are not observed in training data as the results show in Table 11 the proposed model achieved a better detection rate on zero-day attacks.

5.4.5 Additional Comparison

We also compared the proposed model considering the accuracy rate with some of the related works and previously supervised auto-encoder. The evaluation on KDDCUP' 99 also provided in this section; Table 12 shows this comparison in binary classification, and Table 13 provides a comparison in 5 class classification.

All results from 5.4.1 to 5.4.4 show our approach has a better or at least competitive detection rate compared to SVM, SPAE-SVM, related works, and significant potential for real-time NIDS shows in 5.4.1. Comparing our model with related works in binary classification shows our approach has the best accuracy when evaluated on *NSL-KDDTest*$^{+}$, even better than traditional supervised auto-

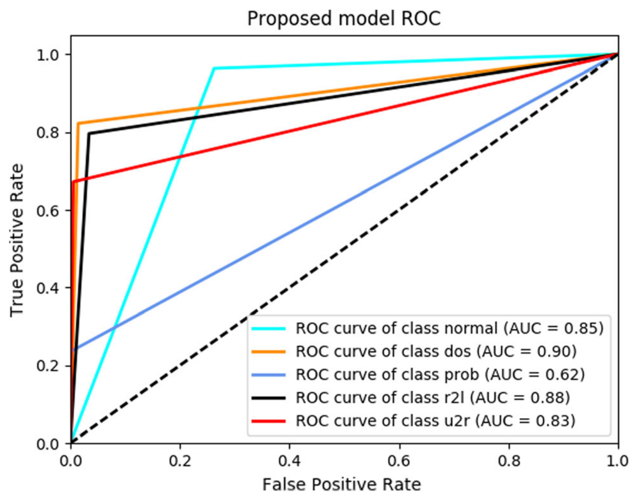

Fig. 6 ROC curve and AUC value of proposed model on five class classification based on NSLKDDTest^{+}

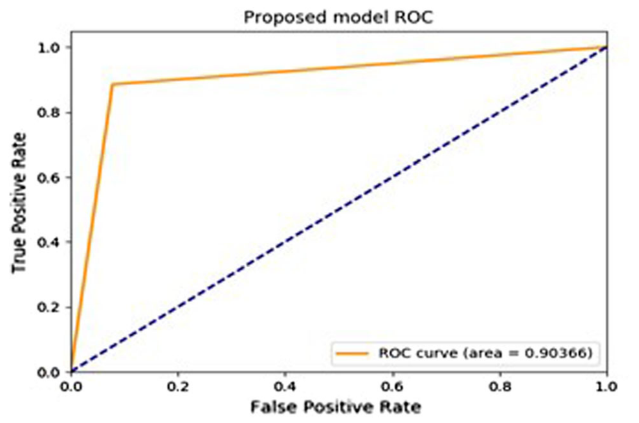

Fig. 7 ROC curve and AUC value of proposed model on binary classification based on NSLKDDTest +

Table 10 Evaluation result considering binary classification on NSL-KDDTrain^{+} data-set

Method	Accuracy	Precision	Recall	*F*-measure
SVM	99.85	99.73	99.68	99.20
SPAE-SVM	99.81	99.68	99.13	99.79
SSPAE-SVM	99.84	99.69	99.92	99.80

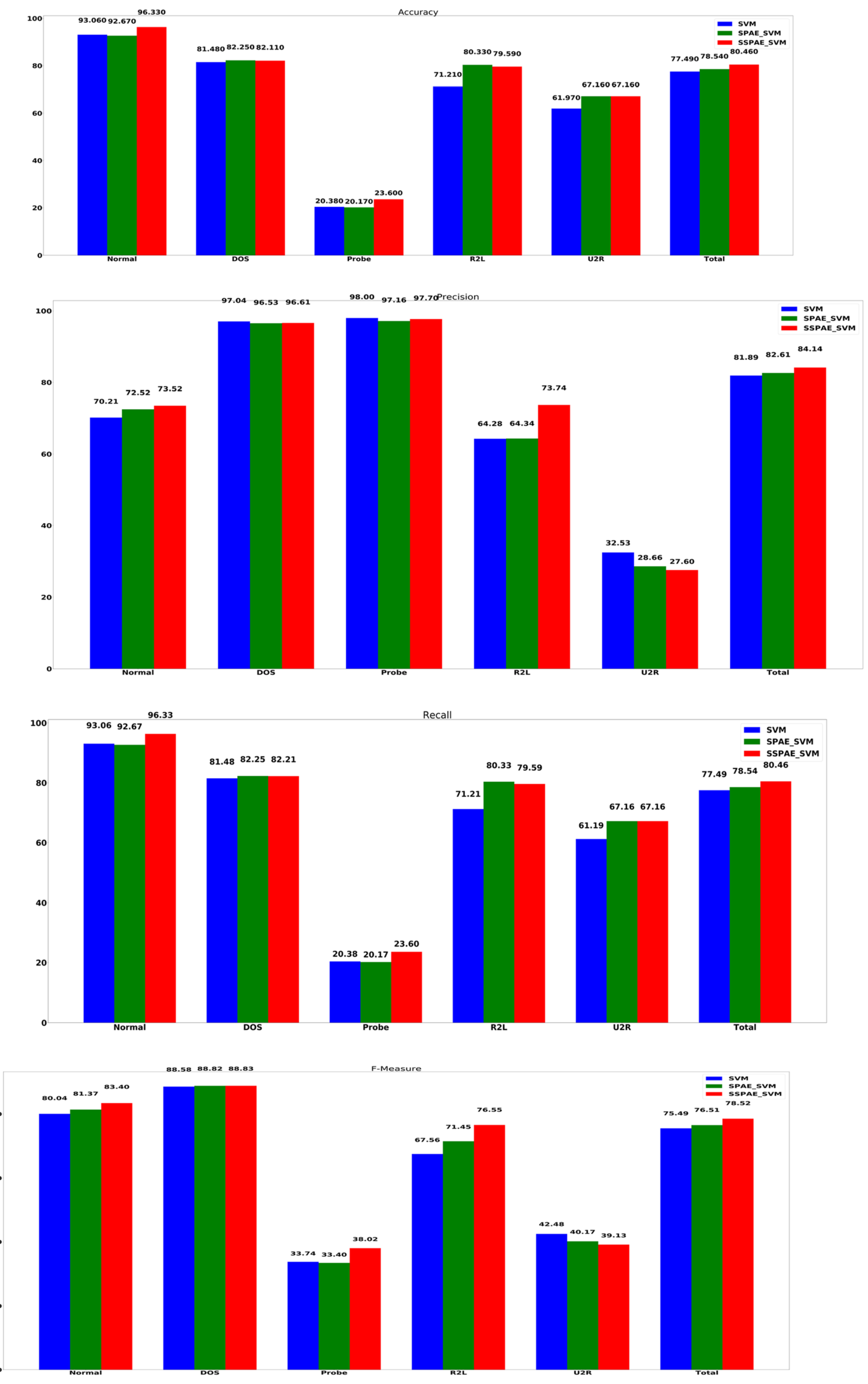

Fig. 8 Accuracy, precision, recall, and *F*-measure values for the proposed model, SVM, SPAE SVM for five class classification based on NSLKDDTest^{+}

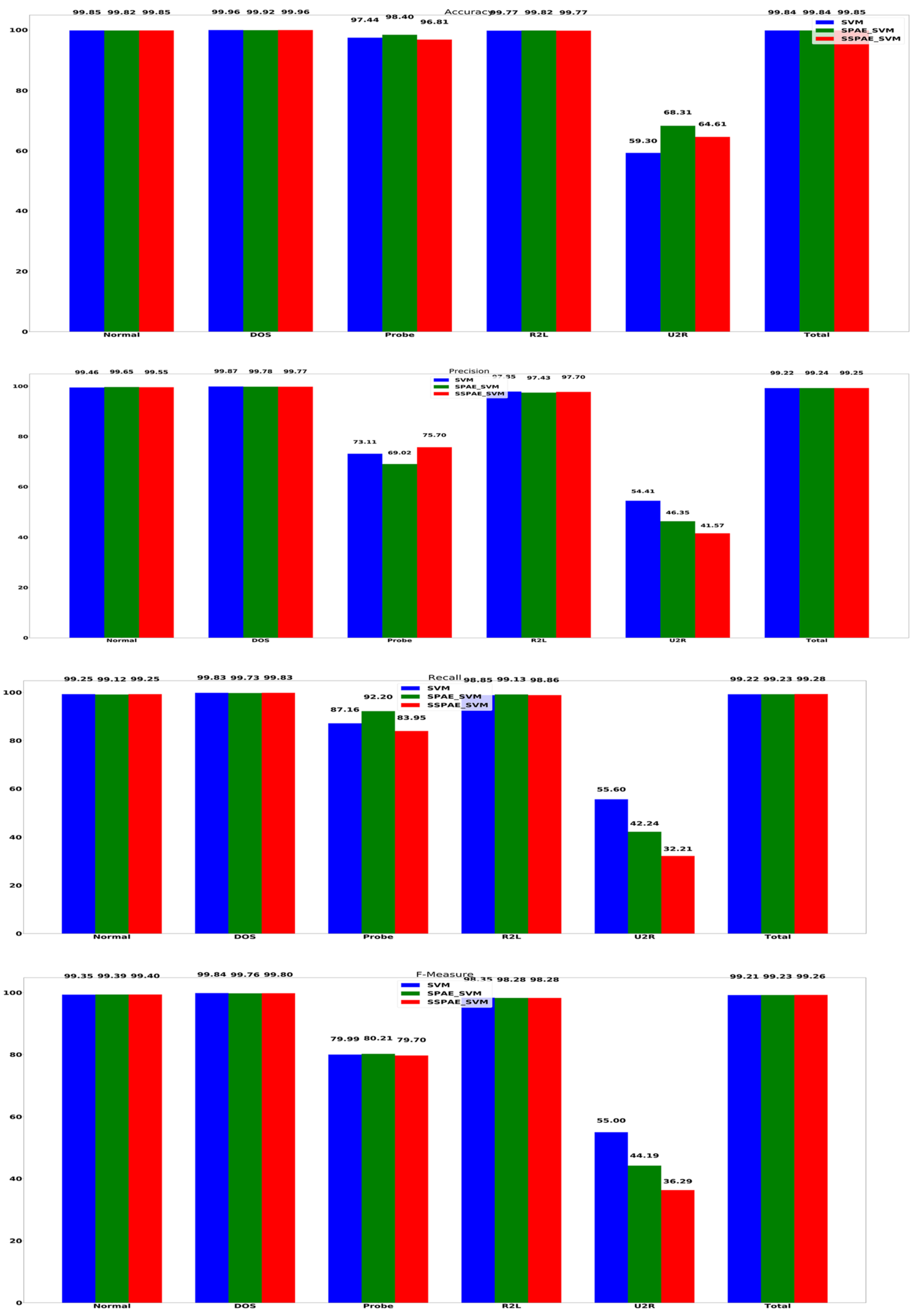

Fig. 9 Accuracy, precision, recall, and *F*-measure values for the proposed model, SVM, SPAE-SVM for five class classification on NSLKDDTrain$^+$

encoder because of these models' high potential overfitting. However, in evaluation on KDD CUP'99 the improvement was just competitive. The reason behind that is the nature of this data set, which is that the corrected and 10% train data set have a close relationship together. If models learn 10% KDD CUP'99 very well, the result will be so good on KDDCUP'99 Corrected, but in *NSL-KDDTrain*$^+$ and *NSL-KDDTest*$^+$ or CICIDS2017, the model like that may face with overfitting problem. Considering improvement through *NSL-KDDTrain*$^+$, *NSL-KDDTest*$^+$, and CICIDS2017 data sets, only the competitive result is 10% KDDCUP'99 and KDDCUP' 99 Corrected shows the

Table 11 Comparison of Zero-day attacks detection rate

Method	The zero-day attacks detection rate	
	Binary (%)	5 class (%)
SVM	78.18	32.08
SPAE-SVM	73.54	33.54
Proposed model	78.34	38.96

robustness of our model on overfitting. The evaluation on *NSL-KDDTest*$^{+}$ with no zero-day attacks also provided for comparison of the proposed model with Shone et al. (2018), considering that their model was much more computationally complex in representation learning by using two deep non-symmetrical auto-encoder with two encoding layers and one decoding layer and also in classification by using random-Forrest (ensemble model) the proposed model achieved better detection rate.

6 Conclusion and Feature Work

In this study, we investigated the challenges of designing a NIDS. In response to these challenges, we proposed a novel supervised sparse auto-encoder with reconstruction loss term weighted by mutual information, which led to significant improvements in both the detection rate and the computational complexity. We have implemented our model in TensorFlow GPU enabled and compared our model with simple SVM and SPAE-SVM for both binary classification and 5class classification in terms of training and test time, accuracy, precision, recall, and f-measure. The improvement of the proposed model in binary classification on NSLKDDTest^{+} with 90.11% accuracy and 91.22 on CICIDS2017 was very promising compared to other works, even much more complex models. Also, in the case of 5class classification, our model has improved compared to SVM and sparse auto-encoder SVM. Moreover, in terms of real-time computation, our model performs better compared to SVM and sparse auto-encoder

Table 12 Additional performance comparisons with several related approaches in the binary classification

Method	Accuracy %	Training data	Testing data	Method category
Naïve Baysian (Tavallaee et al. 2009)	75.56	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Shallow learning
J48 (Tavallaee et al. 2009)	81.07	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Shallow leaning
Random Forest (Tavallaee et al. 2009)	80.67	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Shallow learning
Recurrent Neural Network (Yin et al. 2017)	83.28	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Deep leaning (RNN)
ResNet50 (Li et al. 2017)	79.14	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Deep learning (CNN)
SPAE-Softmax (Javaid et al. 2016)	88.39	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Deep learning (AE)
Deep auto-encoder (Farahnakian and Heikkonen 2018)	96.53	10% KDDCUP'99 (494,021 records)	KDDCUP' 99 Corrected (311,029 records)	Deep learning (AE)
SPAE-SVM (Al-Qatf et al. 2018)	95.09	10% KDDCUP'99 (145,586 records)	KDDCUP' 99 Corrected (77,291 records)	Deep learning (AE) + shallow learning
LightGBM-AE (Tang et al. 2020)	89.82	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Ensemble model + deep learning method
C5 decision tree-one class SVM (Khraisat et al. 2020)	83.24	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Hybrid method
statistical analysis-AE (Ieracitano et al. 2020)	84.21	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Shallow learning
Supervise auto-encoder	82.04	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Deep learning (supervise AE)
Proposed model	90.11	NSL-KDDTrain^{+}	NSL-KDD test^{+}	Deep learning (supervise AE) + shallow learning
Proposed model	94.72	10% KDDCUP'99 (145,586 records)	KDDCUP' 99 Corrected (77,291 records)	Deep learning (supervise AE) + shallow learning

Table 13 Additional performance comparisons with several related approaches in the 5 class classification

Method	Accuracy %	Training data	Testing data	Method category
Naïve Baysian (Yin et al. 2017)	74.40	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Shallow learning
J48 (Yin et al. 2017)	72.80	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Shallow learning
Random Forest (Yin et al. 2017)	75.40	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Shallow learning
RNN (Yin et al. 2017)	81.22	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Deep learning (RNN)
CNN (Wu et al. 2018)	79.48	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Deep learning (CNN)
Deep Belief Net (Gao et al. 2014)	93.94	10% KDDCUP'99 (494,021 records)	KDDCUP' 99 CORRECTED (11,850 RECORDS)	Deep learning (DBN)
Deep Auto-encoder (Farahnakian and Heikkonen 2018)	94.71	10% KDDCUP'99 (494,021 records)	KDDCUP' 99 CORRECTED (311,029 RECORDS)	Deep learning (AE)
Non-symmetrical Auto-encoder + RF (Shone et al. 2018)	85.42	NSL-KDDTrain$^+$	*NSL-KDDTEST*$^+$ WITH NO ZERO-DAY ATTACKS	Deep learning (AE) + shallow learning
SPAE-Softmax(Javaid et al. 2016)	79.10	NSL-KDDTrain$^+$	NSL-KDDTEST$^+$	Deep learning (AE)
Multi-CNN (Li et al. 2020)	81.32	NSL-KDDTrain$^+$	NSL-KDDTEST$^+$	Hybrid-deep learning (CNN)
SPAE-SVM(Al-Qatf et al. 2018)	93.96	10% KDDCUP'99 (145,586 records)	KDDCUP' 99 CORRECTED (77,291 RECORDS)	Deep learning (AE) + shallow learning
Supervised auto-encoder	80.18	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Deep learning (supervised AE)
Proposed model	80.46	NSL-KDDTrain$^+$	NSL-KDDTest$^+$	Deep learning (supervised AE) + shallow learning
Proposed model	88.41	NSL-KDDTrain$^+$	*NSL-KDDTest*$^+$ with no zero-day attacks	Deep learning (supervised AE) + shallow learning
Proposed model	92.24	10% KDDCUP'99 (145,586 records)	KDDCUP' 99 Corrected (77,291 records)	Deep learning (supervised AE) + shallow learning

SVM even with adding an additional computational operator term in the pre-training phase because of faster convergence to the optimal representation.

The comparison of the proposed model with some of the related studies and the traditional supervised auto-encoder has been provided based on accuracy measures for both NSL-KDD and KDDCUP' 99 data sets, all of which were promising. In the future, we aim to provide a model (like using a genetic algorithm) that learns the weights of reconstruction loss instead of using mutual information. We will also try to improve our detection rate on five categories of classification and evaluate the model on real-world data.

Code availability Githhub link for codes: https://github.com/ALIGHORBANI29/IDS_SSPAE_SVM

References

Aburomman AA, Reaz MBI (2016) A novel SVM-kNN-PSO ensemble method for intrusion detection system. Appl Soft Comput 38:360–372

Ahmed M, Mahmood AN, Hu J (2016) A survey of network anomaly detection techniques. J Netw Comput Appl 60:19–31

Ali AH, Shamshirband S, Anuar NB, Petković D (2014) DFCL: dynamic fuzzy logic controller for intrusion detection. Facta Univ Ser Mech Eng 12:183–193

Aljawarneh S, Aldwairi M, Yassein MB (2018) Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. J Comput Sci 25:152–160

Alom MZ, Bontupalli V, Taha TM (2015) Intrusion detection using deep belief networks. In: 2015 National aerospace and electronics conference (NAECON). IEEE, p 339–344

Al-Qatf M, Lasheng Y, Al-Habib M, Al-Sabahi K (2018) Deep learning approach combining sparse autoencoder with svm for network intrusion detection IEEE. Access 6:52843–52856

Alrawashdeh K, Purdy C (2016) Toward an online anomaly intrusion detection system based on deep learning. In: 2016 15th IEEE international conference on machine learning and applications (ICMLA). IEEE, p 195–200

Baldi P (2012) Autoencoders, unsupervised learning, and deep architectures. In: Proceedings of ICML workshop on unsupervised and transfer learning. p 37–49

Aygun RC, Yavuz AG (2017) Network anomaly detection with stochastically improved autoencoder based models. In: 2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud). IEEE, p 193–198

Chapelle O (2007) Training a support vector machine in the primal. Neural Comput 19:1155–1178

Coates A, Ng A, Lee H (2011) An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. p 215–223

Cover TM, Thomas JA (2012) Elements of information theory. Wiley, Hoboken

Deng L, Yu D (2014) Deep learning: methods and applications. Found Trends Signal Process 7:197–387

Eesa AS, Orman Z, Brifcani AMA (2015) A novel feature-selection approach based on the cuttlefish optimization algorithm for intrusion detection systems. Expert Syst Appl 42:2670–2679

Elhag S, Fernández A, Bawakid A, Alshomrani S, Herrera F (2015) On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems. Expert Syst Appl 42:193–202

Farahnakian F, Heikkonen J (2018) A deep auto-encoder based approach for intrusion detection system. In: 2018 20th International conference on advanced communication technology (ICACT). IEEE, p 178–183

Gao N, Gao L, Gao Q, Wang H (2014) An intrusion detection model based on deep belief networks. In: 2014 second international conference on advanced cloud and big data. IEEE, p 247–252

Goodfellow I, Bengio Y, Courville A (2016) Deep learning. MIT Press, Cambridge

Hawkins S, He H, Williams G, Baxter R (2002) Outlier detection using replicator neural networks. In: International conference on data warehousing and knowledge discovery. Springer, p 170–180

Hinton GE, Zemel RS (1997) Minimizing description length in an unsupervised neural network (**Preprint**)

Ieracitano C, Adeel A, Morabito FC, Hussain A (2020) A novel statistical analysis and autoencoder driven intelligent intrusion detection approach. Neurocomputing 387:51–62

Javaid A, Niyaz Q, Sun W, Alam M (2016) A deep learning approach for network intrusion detection system. In: Proceedings of the 9th EAI international conference on bio-inspired information and communications technologies (formerly BIONETICS), ICST (Institute for Computer Sciences, Social-Informatics and…), p 21–26

Khraisat A, Gondal I, Vamplew P, Kamruzzaman J, Alazab A (2020) Hybrid intrusion detection system based on the stacking ensemble of c5 decision tree classifier and one class support vector machine. Electronics 9:173

Kim J, Kim J, Thu HLT, Kim H (2016) Long short term memory recurrent neural network classifier for intrusion detection. In: 2016 International conference on platform technology and service (PlatCon). IEEE, p 1–5

Kingma DP, Welling M (2013) Auto-encoding variational bayes. arXiv preprint arXiv:13126114

Le L, Patterson A, White M (2018) Supervised autoencoders: improving generalization performance with unsupervised regularizers. In: Advances in neural information processing systems. p 107–117

Leung K, Leckie C (2005) Unsupervised anomaly detection in network intrusion detection using clusters. In: Proceedings of the twenty-eighth Australasian conference on computer science, vol 38. Australian Computer Society, Inc., p 333–342

Li W, Yi P, Wu Y, Pan L, Li J (2014) A new intrusion detection system based on KNN classification algorithm in wireless sensor network. J Electr Comput Eng. https://doi.org/10.1155/2014/240217

Li Y et al (2020) Robust detection for network intrusion of industrial IoT based on multi-CNN fusion. Measurement 154:107450

Li Z, Qin Z, Huang K, Yang X, Ye S (2017) Intrusion detection using convolutional neural networks for representation learning. In: International conference on neural information processing. Springer, p 858–866

Liao Y, Vemuri VR (2002) Use of k-nearest neighbor classifier for intrusion detection. Comput Secur 21:439–448

Lin W-C, Ke S-W, Tsai C-F (2015) CANN: An intrusion detection system based on combining cluster centers and nearest neighbors Knowledge-based systems 78:13–21

Louvieris P, Clewley N, Liu X (2013) Effects-based feature identification for network intrusion detection. Neurocomputing 121:265–273

Metz CE Basic principles of ROC analysis. In: Seminars in nuclear medicine, 1978. vol 4. Elsevier, pp 283–298

Mohammed MN, Sulaiman N (2012) Intrusion detection system based on SVM for WLAN. Procedia Technol 1:313–317

Moradi M, Zulkernine M (2004) A neural network based system for intrusion detection and classification of attacks. In: Proceedings of the IEEE international conference on advances in intelligent systems-theory and applications. p 15–18

Nadeem M, Marshall O, Singh S, Fang X, Yuan X (2016) Semi-supervised deep neural network for network intrusion detection. In: KSU proceedings on cybersecurity education, research and practice, pp 1–11

Ng A (2011) Sparse autoencoder. CS294A Lect Notes 72:1–19

Potluri S, Ahmed S, Diedrich C (2018) Convolutional neural networks for multi-class intrusion detection system. In: International conference on mining intelligence and knowledge exploration. Springer, p 225–238

Powers DM (2020) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint arXiv:201016061

Rasmus A, Berglund M, Honkala M, Valpola H, Raiko T (2015) Semi-supervised learning with ladder networks. In: Advances in neural information processing systems, vol 28

Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: ICISSP. p 108–116

Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA (2012) Toward Developing a Systematic Approach to Generate Benchmark Datasets for Intrusion Detection Computers & Security 31:357–374

Shone N, Ngoc TN, Phai VD, Shi Q (2018) A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell 2:41–50

Sindhu SSS, Geetha S, Kannan A (2012) Decision tree based light weight intrusion detection using a wrapper approach. Expert Syst Appl 39:129–141

Smolensky P (1986) Information processing in dynamical systems: Foundations of harmony theory. Colorado Univ at Boulder, Deparment of Computer Science, Boulder

Snort-Network Intrusion Detection & Prevention System (2022). https://www.snort.org. Accessed 18 May 2022

Stolfo SJ, Fan W, Lee W, Prodromidis A, Chan PK (2000) Cost-based modeling for fraud and intrusion detection: results from the JAM project. In: Proceedings DARPA information survivability conference and exposition. DISCEX'00, IEEE, p 130–144

Tang C, Luktarhan N, Zhao Y (2020) An efficient intrusion detection method based on lightGBM and autoencoder. Symmetry 12:1458

Tang TA, Mhamdi L, McLernon D, Zaidi SAR, Ghogho M (2018) Deep recurrent neural network for intrusion detection in sdn-based networks. In: 2018 4th ieee conference on network softwarization and workshops (NetSoft). IEEE, p 202–206

Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the KDD CUP 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, p 1–6

Tsai C-F, Hsu Y-F, Lin C-Y, Lin W-Y (2009) Intrusion detection by machine learning: a review. Expert Syst Appl 36:11994–12000

Valpola H (2015) From neural PCA to deep unsupervised learning. In: Advances in independent component analysis and learning machines. Academic Press, pp 143–171

Viegas E, Santin AO, Franca A, Jasinski R, Pedroni VA, Oliveira LS (2016) Towards an energy-efficient anomaly-based intrusion detection engine for embedded systems. IEEE Trans Comput 66:163–177

Vinayakumar R, Soman K, Poornachandran P (2017) Applying convolutional neural network for network intrusion detection. In: 2017 International conference on advances in computing, communications and informatics (ICACCI). IEEE, p 1222–1228

Wang H, Gu J, Wang S (2017) An effective intrusion detection framework based on SVM with feature augmentation. Knowl Based Syst 136:130–139

Wu K, Chen Z, Li W (2018) A novel intrusion detection model for a massive network using convolutional neural networks. IEEE Access 6:50850–50859

Xiao L, Chen Y, Chang CK (2014) Bayesian model averaging of Bayesian network classifiers for intrusion detection. In: 2014 IEEE 38th international computer software and applications conference workshops. IEEE, p 128–133

Xin Y et al (2018) Machine learning and deep learning methods for cybersecurity. IEEE Access 6:35365–35381

Yin C, Zhu Y, Fei J, He X (2017) A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access 5:21954–21961

Yousefi-Azar M, Varadharajan V, Hamey L, Tupakula U (2017) Autoencoder-based feature learning for cyber security applications. In: 2017 International joint conference on neural networks (IJCNN). IEEE, p 3854–3861

Yu S, Principe JC (2019) Understanding autoencoders with information theoretic concepts. Neural Netw 117:104–123

Zhou Y, Mazzuchi TA, Sarkani S (2020) M-AdaBoost-A based ensemble system for network intrusion detection. Expert Syst Appl 162:113864

Zhu M, Ye K, Xu C-Z (2018) Network anomaly detection and identification based on deep learning methods. In: International conference on cloud computing. Springer, p 219–234