IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 13, 2018, accepted September 5, 2018, date of publication September 13, 2018, date of current version October 12, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2869577

Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection

MAJJED AL-QATF, YU LASHENG, MOHAMMED AL-HABIB, AND KAMAL AL-SABAHI
School of Information Science and Engineering, Central South University, Changsha 410083, China

Corresponding author: Yu Lasheng (yulasheng@csu.edu.cn)

This work was supported by the National Nature Science Foundation of China under Grant Z201610110620003.

ABSTRACT Network intrusion detection systems (NIDSs) provide a better solution to network security than other traditional network defense technologies, such as firewall systems. The success of NIDS is highly dependent on the performance of the algorithms and improvement methods used to increase the classification accuracy and decrease the training and testing times of the algorithms. We propose an effective deep learning approach, self-taught learning (STL)-IDS, based on the STL framework. The proposed approach is used for feature learning and dimensionality reduction. It reduces training and testing time considerably and effectively improves the prediction accuracy of support vector machines (SVM) with regard to attacks. The proposed model is built using the sparse autoencoder mechanism, which is an effective learning algorithm for reconstructing a new feature representation in an unsupervised manner. After the pre-training stage, the new features are fed into the SVM algorithm to improve its detection capability for intrusion and classification accuracy. Moreover, the efficiency of the approach in binary and multiclass classification is studied and compared with that of shallow classification methods, such as J48, naive Bayesian, random forest, and SVM. Results show that our approach has accelerated SVM training and testing times and performed better than most of the previous approaches in terms of performance metrics in binary and multiclass classification. The proposed STL-IDS approach improves network intrusion detection and provides a new research method for intrusion detection.

INDEX TERMS Network security, network intrusion detection system, deep learning, sparse autoencoder, SVM, self-taught learning, NSL-KDD.

I. INTRODUCTION

Approximately 50 billion devices are expected to be connected to the Internet by 2020 due to the wide range of communication and network technologies that have changed our daily lives. These technologies have been used worldwide in nearly all organizational operations, such as online shopping, banking, industrial applications, and email systems. Although the benefits provided by these technologies have improved our lives and changed the world, information security remains a crucial issue. Organizations need to provide secure communication channels to Internet users, including the organizations' customers and employees, and detect unauthorized activities. Currently, network intrusion detection systems (NIDS) offer a better solution to the security problem compared with other traditional network defense technologies, such as firewall systems. NIDS helps network administrators detect attacks, vulnerabilities, and breaches inside an organization's network. The two forms of NIDS are signature-based NIDS (SNIDS) and anomaly detection-based NIDS (ADNIDS). In SNIDS, the system detects attacks on the basis of rules that are pre-installed for attacks in NIDS. Network traffic is compared with an updated database of attack signatures to detect intrusion in the network traffic dataset.

In ADNIDS, the system classifies unknown or unusual behavior in network traffic by studying the structures of normal behavior in network traffic. Network traffic that deviates from a normal traffic pattern is classified as an intrusion. The advantage of ADNIDS is that unknown/new attacks can be predicted. Therefore, we focus on this type of intrusion detection systems. Anomaly detection methods can be used in various areas, such as network security, fraud detection in

credit cards, military applications, and many medical applications [1]. The following can be performed to develop an effective anomaly-based intrusion detection system. First, proper feature selection based on feature extraction and dimensionality reduction must be implemented when extracting a subset of a correlated features from the network traffic dataset to enhance classification results [2]. Second, the best techniques must be used to help enhance the classification results and increase the classification speed.

Various supervised and unsupervised machine learning techniques can be used or integrated with other algorithms in ADNIDS to enhance intrusion detection performance and increase the classification rate of machine learning algorithms, such as decision tree, random forest, self-organization maps (SOM), and support vector machine (SVM), which have been utilized to detect and classify intrusions. Many researchers that investigated NIDS focused on using unsupervised learning techniques followed by shallow machine learning, such as SVM, random forest, and naïve Bayesian [3], [4], because using unsupervised learning techniques before shallow machine learning offers improvements in detection rate. Unsupervised learning algorithms and dimension reduction methods are frequently used in feature extraction and feature representation to improve data quality [5] and achieve an improvement in the classification results of shallow and traditional supervised machine learning algorithms. Recently, remarkable achievements in unsupervised deep learning-based methods were successfully applied in vision computing applications. In addition, deep learning techniques [6]–[10] can be adopted as unsupervised feature learning methods that help supervised machine learning improve its performance and identification of network traffic anomalies by reducing the testing and training times.

Deep learning approaches have a good potential to achieve effective data representation for building improved approaches. Therefore, on the basis of a self-taught learning framework and inspired by the combination of the sparse autoencoder (SAE) with SVM, we propose using self-taught learning (STL) for good data representation and SVM for the classification task. STL is a deep learning approach that is based on the SAE algorithm. It helps rebuild input representation and converts it to feature representation of data related to the input data, thereby improving the performance of the classification task considerably. The main contributions of this work are as follows:

(1) We develop a novel deep learning approach **STL-IDS** (a self-taught learning based intrusion detection system) based on the STL framework by combining SAE and SVM for network intrusion detection. We study the potential of our approach to achieve effective representation and dimensionality reduction for the improvement of the classification results of shallow and traditional supervised machine learning algorithms, such as SVM, in binary and multiclass classification.
(2) We combine deep and shallow learning techniques in our novel approach and exploit their respective strengths. Better or at least competitive results are achieved compared with the results of similar approaches. Moreover, our approach considerably reduces training and testing times of SVM.
(3) We use the NSL-KDD dataset to compare the efficiency of our approach with single SVM and that of different classification algorithms, such as naïve Bayesian, random forest, multi-layer perceptron, and many other classification algorithms in related work on binary and multiclass classification.

Experimental results show that our approach is suitable for intrusion detection. Its performance is superior to that of traditional classification algorithms using the NSL-KDD dataset and most previous approaches in terms of binary and multiclass classification.

The rest of this paper is structured as follows. We briefly describe related studies, particularly those that examined unsupervised machine learning techniques and SVM-based deep learning approaches, in Section II. In Section III, we provide an overview of our proposed methodology for NIDS implementation, SVM, the NSL-KDD dataset, data processing, and evaluation metrics. In Section IV, the performance of our approach is evaluated based on the experimental results and compared with that of related approaches for NIDS. Our conclusions and directions for future research are presented in Section V.

II. RELATED WORK

Network intrusion detection has become the most important part of the infrastructure of defense networking systems in information security. Various machine learning algorithms or approaches are applied in NIDS to detect and distinguish between normal traffic and anomalies or attacks in network traffic; these approaches include decision tree [11], k-nearest neighbor (K-NN) [12], naïve Bayes network [13], [14], SOM [15], [16], SVM, and artificial neural network (ANN) [17]. SVM demonstrates better performance than other traditional machine learning classification techniques [18].

A work proposed by Mukkamala *et al.* [19] compared the performance of SVM and ANN on the KDD CUP 99 dataset. The results showed that the detection results of SVM are better than those of ANN. In [20], SVM, naïve Bayes, logistic regression, decision tree (DT), and classification and regression tree (CART) approaches were compared in terms of intrusion detection classification by using the KDD CUP 99 dataset. The results showed that SVM has distinct features. Ashfaq *et al.* [21] developed a new method by using the fuzziness approach based on semi-supervised learning for intrusion detection. This method uses a neural network with random weights and plays an important role in the detection rate of NIDS because it decreases the computational cost. The model was evaluated on the NSL-KDD dataset but the performance of the model was studied on only the binary classification task. In [22], a deep learning model based on a recurrent neural network with a soft-max classifier

was presented. The model was evaluated on the NSL-KDD dataset, and the performance of the model in binary and multiclass classification was studied. The model showed a deep learning capability to model high-dimensional features, and an improved accuracy rate of intrusion detection was achieved. However, the training time was large. SVM is one of the most important traditional machine learning algorithms that depend on statistical learning theory, and it uses structural risk minimization to achieve a strong generalization capability. In addition, SVM presents a constrained quadratic programming problem that requires a large memory and considerable training time. In addition, the training complexity of SVM is highly dependent on the size of the dataset. Therefore, the performance of SVM based on IDS needs to be enhanced, and the training and testing times must be reduced. To address these limitations, many studies have improved SVM-based IDS by combining SVM with other methods. In [23], efficient machine learning based on SVM with feature augmentation was presented to increase the quality of the SVM classifier. The method improved the intrusion detection rate of SVM and reduced the required training time. The weakness of this approach is that its detection accuracy is insufficient, and the time factor are not considered.

Many researchers have combined supervised and unsupervised learning algorithms to create a model that can increase the detection rate of supervised machine learning classifiers, such as an SVM and random forest. In [24], many unsupervised learning algorithms were combined with SVM and a neural network (NN) to improve the performance of the intrusion detection system. The authors designed, implemented, and evaluated many hybrid models that use principal component analysis (PCA) or Gradual Feature Reduction (GFR) for feature selection and SVM or NN for classification. The results showed that hybrid models can effectively detect known and unknown attacks, and PCA and GFR feature selection techniques are computationally expensive in terms of training and testing times. Unsupervised learning based on deep learning has been used recently in feature extraction and dimensionality reduction, leading to an increase in the detection rate and a decrease in the processing time of supervised machine learning algorithms, such as SVM and soft-max. Alom *et al.* [8] proposed a deep learning approach based on stack restricted Boltzmann machine for feature extraction and dimensionality reduction and based on SVM for the classification of network intrusion detection. The approach was implemented on merely 40% of the NSL-KDD training dataset. The approach performed better than single SVM or single deep belief networks (DBN) and many other approaches. In [10], a feature learning model based on AE was presented to achieve a good representation of different feature sets. This feature learning model was applied to malware classification and anomaly-based network intrusion detection by using the NSL-KDD dataset. The topology of the used AE was different from the common topology, and the extracted feature by the AE was applied to many traditional machine learning algorithms, such as SVM, K-NN, and Gaussian naïve Bayes. The experimental results showed that the model is an improvement of traditional machine learning. However, the model is computationally expensive because it comprises many hidden layers and entails two training stages. Diro and Chilamkurti [25] presented IOT/fog network attack detection system. Their system was based on distributed deep learning. The performance of their model was compared against shallow and traditional machine learning approaches. They used deep learning model with three hidden layers for feature learning and soft-max regression (SMR) for the classification task. However, their model is computationally expensive compared to our approach. Their model was evaluated on NSL-KDD dataset in both binary and multiclass classification. Their method has demonstrated that the distributed attack detection can better detect attacks than centralized detection system because of the parameter sharing that can avoid local minima in training. Our approach has a significant difference since we aimed at adopting a new approach to enable the detection of attacks through centralized detection system. By contrast, their model aimed at adopting new approach to enable the detection of the attacks through distributed detection system in social internet of things.

Wang *et al.* [26] proposed novel intrusion detection system called hierarchical spatial-temporal feature-based intrusion detection system (HAST-IDS); their system used the deep convolutional neural network for learning the low-level spatial features of network traffic, LSTM networks (long short-term memory networks) for learning high-level temporal features. They used the standard DRAPA and ISCX2012 dataset to evaluate the performance of their proposed system. Their model is computationally expensive compared to our approach because they used two stages for feature learning. Farahnakian and Heikkonen [27] proposed deep learning approach for intrusion detection. The model was built using deep autoencoder and trained in a greedy layer wise fashion in order to avoid overfitting. The performance of their approach was evaluated on KDDCup 99 (old version of NSL-KDD); their approach performance was studied on both binary and multiclass classification. Although high accuracy was achieved for intrusion detection task, their approach is computationally expensive compared to our approach.

Madani and Vlajic [28] have studied the viability of using the deep autoencoder in anomalies detection in adaptive intrusion detection system under adversarial contamination. They used the reconstruction error of the autoencoder as a measure for anomaly detection and the NSL-KDD dataset for performance evaluation. Our approach is significantly different since we used the autoencoder for feature learning and dimensionality reduction. Moreover, we used the performance metrics, training time, and testing time to evaluate the performance of our model for anomaly detection.

In [2], NIDS based on unsupervised deep learning techniques was developed using SAE for feature learning and soft-max regression (SMR) for classification. Evaluation was based on all performance metrics on the NSL-KDD

dataset using the two evaluation approaches in Section IV: KDDTrain+ using 10-fold cross-validations for two-category (normal traffic and attacks) and five-category (normal and five types of attacks) classification. Furthermore, evaluations were performed separately for training and testing based on KDDTrain+ and KDDTest+. The evaluation results showed that the approach shows better performance in terms of accuracy rate for two-category classification compared with SMR and many other approaches. The weakness of this approach is that the dimensionality reduction mechanism, which significantly reduces the training and testing times for intrusion detection, is not considered. In addition, both evaluation approaches obtained low accuracy in the five-category classification. Although many unsupervised learning-based network intrusion detection methods have been presented in recent years, several of them still suffer from limitations and issues, such as the following:

- Shallow learning is inappropriate for intelligent analysis, and the predicting requirements of high-dimensional learning have redundant features. Hence, a raw dataset leads to reduced classifier accuracy. By contrast, deep learners can achieve an effective representation, thus improving the classification results of shallow and traditional supervised machine learning algorithms, such as SVM and RF.
- Although several approaches that depend on deep learning techniques are effective, they continue to suffer from time complexity.

On the basis of this analysis, we propose a combination of SAE and SVM based on STL. First, we use SAE for effective representation of our raw dataset (NSL-KDD), followed by the use of SVM for classification. The accuracy rate of SVM and training and testing times are optimized simultaneously.

III. PROPOSED METHODOLOGY: SAE–SVM AND NSL-KDD DATASET OVERVIEW

A. PROBLEM FORMULATION

In our self-taught learning approach (SAE–SVM), we are given a labeled training set of m records $\left\{\left(x_l^{(1)}, y^{(1)}\right), \left(x_l^{(2)}, y^{(2)}\right), \ldots, \left(x_l^{(m)}, y^{(m)}\right)\right\}$, where input feature vector $x_l^{(i)} \in \mathbb{R}^n$ (The subscript "l" indicates that it is a labeled record), $y^{(i)} \in \{+1, -1\}$ are the corresponding labels for binary classification, $y^{(i)} \in \{1, 2, \ldots., C\}$ are corresponding labels for multiclass classification. Additionally, we assume there are m unlabeled samples $x_u^{(1)}, x_u^{(2)}, \ldots., x_u^{(\mathrm{m})} \in \mathbb{R}^n$ produced by removing the labels from the labeled training set. For a better representation and less dimensionality of the input training set $x_l^{(1)}, x_l^{(2)}, \ldots, x_l^{(\mathrm{m})} \in \mathbb{R}^n$, as in Figure 1-STEP1, we feed the unlabeled sample $x_u^{(1)}, x_u^{(2)}, \ldots., x_u^{(\mathrm{m})} \in \mathbb{R}^n$ (KDDTrain+) to the sparse autoencoder algorithm. It can be used to reconstruct and learn the input training dataset $x_l^{(1)}, x_l^{(2)}, \ldots, x_l^{(\mathrm{m})} \in \mathbb{R}^n$. After learning the optimal values for W and b_1 (Trained parameter set in Figure 1) by applying SAE on unlabeled data $\boldsymbol{x_u}$(KDDTrain+), as in

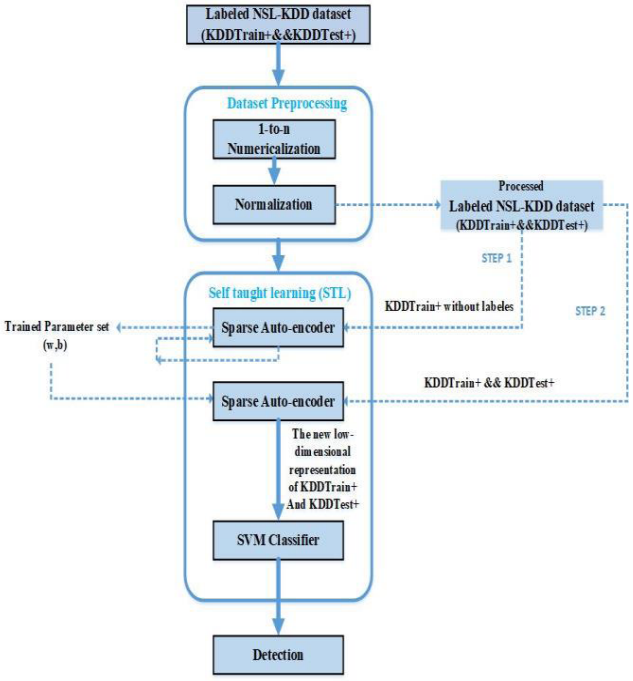

FIGURE 1. Block diagram of the proposed STL-IDS.

Figure 1-STEP2, we feed $x_l^{(1)}, x_l^{(2)}, \ldots, x_l^{(\mathrm{m})}$ (KDDTrain+ and KDDTest+ dataset) as an input to a sparse autoencoder which attempts to reconstruct and learn its output values $(\widehat{x_l}^{(1)}, \widehat{x_l}^{(2)}, \widehat{x_l}^{(3)}, \ldots., \widehat{x_l}^{(m)})$ to be equal to its inputs $(x_l^{(1)}, x_l^{(2)}, \ldots., x_l^{(\mathrm{m})})$, getting a new and good representation $\left\{\left(h_l^{(1)}, y^{(1)}\right), \left(h_l^{(2)}, y^{(2)}\right), \ldots\ldots, \left(h_l^{(m)}, y^{(\mathrm{m})}\right)\right\}$ where the original input data is replaced with corresponding vector of activations h as in Figure 2. Thus, our training set becomes $\left\{\left(h_l^{(1)}, y^{(1)}\right), \left(h_l^{(2)}, y^{(2)}\right), \ldots\ldots, \left(h_l^{(m)}, y^{(\mathrm{m})}\right)\right\}$,. Finally, we train SVM using the new training set to obtain a function that performs predictions of the intrusion on the y values. For the given testing set x_{test}, we follow the same scenario for the training set: feeding it to sparse autoencoder to get h_{test}. Then, we feed h_{test} to the trained SVM classifier to get a prediction. Our goal is to improved SVM classification accuracy and accelerating the training and testing and to develop a network intrusion detection model that can accurately and quickly predict the intrusions in both binary and multiclass classification on NSL-KDD dataset and the pre-learned sparse autoencoder with SVM. The detailed steps of the proposed approach will be presented in the next subsections. Our STL model based on SAE and SVM involves many steps (Figure 1). The basic methodology is as follows.

B. STL: SAE–SVM

STL [29] is a new deep learning framework that involves two stages. In the first stage, new effective representation is obtained from our NSL-KDD dataset without label, $\boldsymbol{x_u}$, and is called unsupervised feature learning (UFL). The new

representation is related to unlabeled data. In the second stage, the new representation is combined with labeled data, $\boldsymbol{x_l}$, then any supervised algorithm, such as SVM, can be used for the classification task.

Different methods can used for UFL [30]; for our model, we adopt SAE which is an unsupervised learning algorithm that consists only of a single hidden layer. It can be used for feature learning and dimensionality reduction instead of PCA to achieve a significantly nonlinear generalization. Its input and output layers have the same number of units. The input and output layers contain N units, and the hidden layer contains K units. As shown in Figure 2(a), the output values $\widehat{x_i}$ in the output layer is similar to the input values x_i in the input layer.

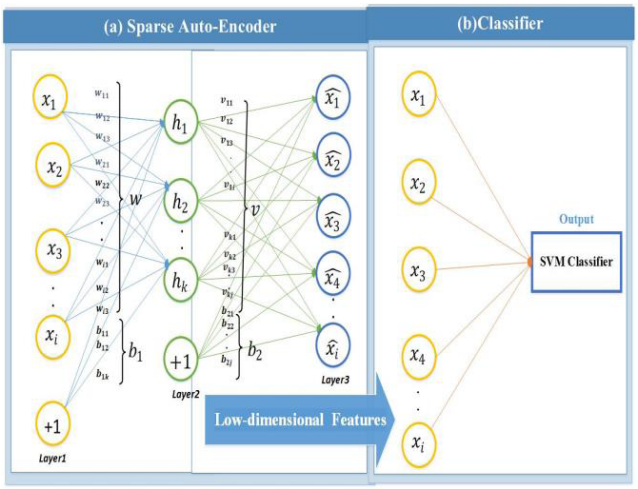

FIGURE 2. Self-taught learning stages.

In the recent years, there is increasing attention to the study of single-layer SAE as a feature learning and dimensionality reduction method. The SAE can learn effective low-dimensional features from the raw data and make it easier to extract efficient and appropriate low-dimensional features automatically for the classification process.

Feature extraction and dimensionality reduction process in SAE involves two steps: encoding and decoding. The encoding step maps the input data x_i into the hidden units' representations, as shown in (1a):

$$h = f(X) = g(WX + b_1) \tag{1a}$$

The encoding step maps the hidden units' representations into the reconstructed data, as shown in (1b):

$$\mathrm{Z} = g\left(\mathrm{V}h + b_2\right) \tag{1b}$$

In the above equations, $X = (x_1, x_2, x_3, x_4, \ldots, x_i)$ is the high-dimensional input data vector,$Z = (\widehat{x_1}, \widehat{x_2}, \widehat{x_{3,\ldots\ldots}}, \widehat{x_m})$ is the reconstruction vector of the input data and $\mathrm{h} = (h_1, h_1, h_1, \ldots, h_k)$ is the low-dimensional vector output from the hidden layer.

SAE applies backpropagation algorithm to obtain the optimal values for its weight matrices $\mathrm{W} \in \Re^{K\times N}$ and $\mathrm{V} \in \Re^{N\times K}$ and bias vectors $b_1 \in \Re^{K\times 1}$ and $b_2 \in \Re^{N\times 1}$, which attempt to learn and reconstruct its output values $\widehat{x_i}$ to be equal to its inputs x_i. In other words, an approximation to the identity function is learned to make the output values similar to the input values; that is, it uses $y^{(i)} = x^{(i)}$ [3], [31]. The activation function is chosen to be the sigmoid function, $g\,(z) = \frac{1}{1+e^{-z}}$, and its output range is [0,1]. It is used for the activation (hW, b) of the nodes in the hidden and output layers are shown in (1a).

$$\begin{aligned}\mathcal{T} = {} & \frac{1}{2m}\sum_{i=1}^{m} \| x_i - \widehat{x_i} \|^2 \\ & + \frac{\lambda}{2}\left(\sum_{k,n} W^2 + \sum_{n,k} V^2 + \sum_{k} b_1^2 + \sum_{n} W{b_2}^2\right) \\ & + \beta \sum_{j=1}^{k} KL(\rho \parallel \widehat{p_j})\end{aligned} \tag{2}$$

SAE also applies backpropagation to minimize the cost function, which is represented by Eq.2 [2]. The first term is the average sum-of-square errors for all m input data. The second term is a weight decay parameter (λ) used for tuning the weights between the hidden and output units to improve performance and prediction while helping check and avoid overfitting. The last term in the equation is the sparsity penalty term that places a constraint on the hidden layer to maintain low average activation values; it is expressed as Kullback–Leibler (KL) divergence shown in Eq. 3 [31].

$$KL\left(\rho \parallel \widehat{p_j}\right) = \rho \log \frac{\rho}{\widehat{\rho_j}} + (1-\rho)\log \frac{1-\rho}{1-\widehat{p_j}}, \tag{3}$$

where ρ is a sparsity constraint parameter that ranges from 0 to 1 and β controls the sparsity penalty term. $KL\left(\rho \parallel \widehat{p_j}\right)$ attains a minimum value when $\rho = \widehat{p_j}$, where denotes $\widehat{p_j}$ the average activation value of hidden unit j over all training inputs x. After learning the optimal values for W and b_1 by applying SAE on unlabeled data $\boldsymbol{x_u}$, we evaluate the feature representation $a = \mathrm{h}$ for labeled data (x_l, y). We use this new feature representation, h, with the label vector, y, in SVM for the classification task in the second stage of STL, as shown in Figure 2(b). Figure 2 shows an architectural diagram of the proposed STL. We apply STL based on SAE for good data representation because of its simple and straightforward implementation and its capability to learn the original expressions and structures of data. The wide application of STL extends particularly to image identification [32], [33], SVM for classification tasks, and distinguishing different types of intrusions because combining robust classifiers, such as SVM, with SAE leads to enhanced performance in intrusion detection. Furthermore, the features extracted from the SAE algorithm are passed to the SVM classifier for intrusion detection. The performance accuracy rate of our method is better than that of SVM alone, and the training and testing times of SVM are reduced.

C. SVM

The SVM classifier relies on statistical learning theory (SLT) and produces a hyperplane to isolate a class of positive instances from a class of negative instances by using structural risk minimization rules. SVM aims to split data points with a hyperplane and determine which class each data point belongs to. SVM maximizes the margin between support vectors because separating all classes is necessary [34]. SVM is a popular learning technique due to its high classification accuracy and performance in solving regression and classification tasks. SVM was initially designed for binary classification. Later, it was extended to multi-class scenarios. The many basic functions of SVM include linear, polynomial, sigmoid, and RBF kernels. We use the RBF kernel, which is also known as the Gaussian kernel, in our research. RBF has two parameters: C and σ. Both can be artificially adjusted, and different parameter values correspond to the nature of classifiers. The performance of SVM depends on selecting suitable kernel function types and proper parameters of the kernel function for our problem. In our proposed approach, we use the automatic parameter selection method by applying k-fold cross-validation (CV) to search for the best parameter of the RBF kernel. Many strategies, such as one versus the rest (OvsR) and one versus one (OvsO), can be used to build a multi-class SVM classifier. We use the OvsR strategy in our method. Given that SVM consumes much time for training, numerous approaches are implemented in SVM to reduce the required processing time for classification and prediction tasks. The storage requirements and computational complexity of the SVM with RBF kernel depend on both input dimensionality (d) and the number of support vectors (nSV). Generally, the storage requirements and computational complexity is bound by O(d nSV).

D. NSL-KDD DATASET OVERVIEW

The NSL-KDD dataset was recommended in 2009 by Travallaee *et al.* [35] because of the inherent drawbacks of KDD CUP99 [35]. Subsequently, many researchers in intrusion detection used

NSL-KDD to evaluate their approaches, similar to what was done in [21] and [36]. NSL-KDD was built based on the KDD CUP 99 dataset, but the redundant instances were removed and the structure of the dataset was reconstituted [35]. The NSL-KDD dataset is normally used to evaluate the effectiveness of proposed approaches for intrusion detection, especially anomaly-based network intrusion detection. NSL-KDD has a reasonable number of records in training and testing sets. The total number of records in the training set (KDDTrain+) is 127,973, and the testing set (KDDTest+) has 22,544 records. Each traffic record in the NSL-KDD dataset contains 41 features (6 symbolic and 35 continuous), as shown in Table 1, and 1 class label. The features can be categorized into three types: basic, content, and traffic (Table 1). According to feature characteristics, the attacks in the NSL-KDD dataset can be classified into four types: user-to-remote attacks (U2R), denial of service attacks (DOS), root-to-local attacks (R2L), and probing attacks (Probe). These types and categories are summarized in Table 2. Several attacks exist in the testing set (KDDTest+) but not in the training set (KDDTrain+). The difference between training and testing sets provides a highly realistic theoretical basis for intrusion detection.

TABLE 1. Attack types and categories.

Attack type	Attack categories
Dos	Back, Land Neptune, Pod, Smurf, Teardrop, Apache 2, Udpstorm, Process table, Worm (10)
Probe	Satan, Ipsweep, Nmap, Portsweep, Mscan, Saint (6)
R2L	Guess_Password, Ftp_write, Imap, Phf, Multihop,Waremaster, Warezclient, Spy, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel, Sendfmail, Named (16)
U2R	Buffer_overflow, Loadmodule, Rotkit, Perl,Sqlattack, Xterm, Ps (7)

E. DATA PREPROCESSING

1) 1-TO-N NUMERICAL ENCODING

The SAE–SVM algorithm used in our approach cannot directly process the NSL-KDD dataset in its original format. However, we use a 1-n encoding system to convert non-numeric features into numeric features before applying STL, as shown in Figure 1. The NSL-KDD dataset has three non-numeric features and 38 numeric features. Hence, we apply a 1-n encoding system to the non-numeric features, such as ''protocol-type'' ''service,'' and ''flag,'' as follows:

(1) We convert the ''protocol-type'' feature into a numeric feature. The ''protocol-type'' feature has three distinct attributes, namely, tcp, udp, and icmp, and these can be encoded as (1,0,0), (0,1,0), (0,0,1) in binary vectors, respectively.

(2) We convert the ''service'' and ''flag'' features into numeric features. The ''service'' feature has 70 distinct attributes, and the ''flag'' feature has 11 distinct attributes. By using the same method in the first step, each distinct attribute of ''service'' is mapped into 70-dimensional binary attributes, and each distinct attribute of ''flag'' is mapped into 11-dimensional binary attributes. After all the transformations, the 41-dimensional features of the NSL-KDD dataset are mapped into 122-dimensional features.

2) NORMALIZATION

Several of the features of the NSL-KDD dataset have very large ranges between the maximum and minimum values, such as the difference between the maximum and minimum values in ''duration [0, 58329],'' where the maximum value is 58,329 and the minimum is 0. A large difference also exists in other feature values, such as ''src-bytes'' and ''dst-bytes,''

TABLE 2. Feature details of the NSL-KDD dataset.

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

thereby making the feature values incomparable and unsuitable for processing. Hence, these features are normalized by using max–min normalization for mapping all feature values to the range [0, 1] according to Eq. (4).

$$x_i = \frac{x_i - Min}{Max - Min}, \tag{4}$$

Where x_i denotes each data point, ***Min*** denotes the minimum value from all data points, and ***Max*** denotes the maximum value from all data points for each feature.

F. EVALUATION METRICS

We use NSL-KDD (KDDTrain+ and KDDTest+) to verify the superiority of our approach in improving the SVM classification results for network intrusion detection. All performance metrics are used to measure the performance of our proposed approach. The attribute values that resulted from the training and testing processes of the NSL-KDD dataset are used to calculate these performance metrics. The values can be defined as follows:

- True positive (TP): anomaly instances correctly classified as an anomaly.
- False positive (FP): normal instances wrongly classified as an anomaly.
- True negative (TN): normal instances correctly classified as normal.
- False negative (FN): anomaly instances wrongly classified as normal.

Then, we compute the performance metrics from the following notations.

- Accuracy (AC): indicates the proportion of correct classifications of the total records in the testing set, as shown in (5).

$$AC = \frac{TP + TN}{TP + TN + FP + FN} \tag{5}$$

- Precision (P): indicates the proportion of correct predictions of intrusions divided by the total of predicted intrusions in the testing process, as shown in (6).

$$p = \frac{TP}{TP + FP} \tag{6}$$

- Recall (R): indicates the proportion of correct predictions of intrusions divided by the total of actual intrusion instances in the testing set, as shown in (7).

$$R = \frac{TP}{TP + FN} \tag{7}$$

- F-measure (F): is considered the most important metric of network intrusion detection that represents both precision (P) and recall (R), as shown in (8).

$$F = \frac{2 * P * R}{P + R} \tag{8}$$

IV. PERFORMANCE EVALUATION: IMPACT OF THE LOW-DIMENSIONAL FEATURES AND DIFFERENT HIDDEN UNITS AND SPARSITY PARAMETER ON SVM CLASSIFIER

Experiments are performed on a PC with Intel(R) Core(TM) i5-6400 CPU at 2.71GHZ with 8 GB of RAM and running on Windows 10. Our approach was implemented in MATLAB, and the SVM classifier is applied with the LIBSVM package [37] (MATLAB version 3.22). Our dataset is processed in python language. The RBF kernel is used as the SVM classifier, and k-fold cross-validation is applied to search for the best parameter of the RBF kernel. The performance evaluation of our approach based on the NSL-KDD dataset is performed in two ways as follows:

- Training (KDDTrain+) and testing (KDDTest+) data are used separately for training and testing.
- Ten-fold cross-validation is performed on KDDTrain+ for training and testing.

Our experiments are conducted to study the performance efficiency and verify the effectiveness of the low-dimensional features extracted by our approach for binary (normal, anomaly) and multiclass (normal, DoS, R2L, U2R, and Probe) classification based on the NSL-KDD dataset. Furthermore, the training and testing times are calculated to evaluate the efficiency of our model. In addition, we also focused on addressing intrusion detection system requirements that have faster and lower computational costs by reducing computational complexity and storage requirements of SVM classier. To achieve these requirements, we focused on the extraction of the good data representation and low-dimensional features from raw data, feeding it into SVM classifier for reducing the number of support vectors(nSVs) of SVM because non-linear kernels require memory and computation that grow linearly proportional to the SVs [38]. Generally, the storage requirements and computational complexity of SVM grows linearly proportional to the number of SVs they have [39].

Thus, because the storage requirements and computational complexity of SVM with RBF kernel depend on both input dimensionality (d) and the number of support vectors (nSV), as discussed in Section III-C, our model has reduced the storage requirements and computational complexity of SVM compared to SVM alone because its capability to achieve low-dimensional representation from raw data and less support vector number of SVM need to be stored as shown in the Table 3-Column nSV, Table 3-Column nSV, Table 4-Column nSV, Table 5-Column nSV, Table 6-Column nSV.

TABLE 3. Training and testing time and number of support vectors (NSV) comparison for STL-IDS and single SVM for binary classification based on testing data.

Method	Training Time(sec)	Testing Time(sec)	nSV
Single SVM	900.075	19.934	3649
STL-IDS	673.031	4.648	3448

TABLE 4. training and testing time and number of support vectors (NSV) comparison for STL-IDS and single SVM for binary classification based on training data.

Method	Training Time(sec)	Testing Time(sec)	nSV
Single SVM	7754.810	101.224	33988
STL-IDS	5929.533	21.304	30096

TABLE 5. Training and testing time and number of support vectors (NSV) comparison for STL-IDS and single SVM for five-category classification based on testing data.

Method	Training Time (sec)	Testing Time (sec)	nSV
Single SVM	3503.118	63.222	11557
STL-IDS	1362.384	5.395	7790

Finally, we compare the performance of our approach with that of existing methods, such as naive Bayesian, RF, multi-layer perceptron, SVM, and shallow machine learning, as mentioned in [22] and [35] and several recent approaches.

A. EVALUATION THE IMPACT OF THE LOW-DIMENSIONAL FEATURES ON THE BINARY CLASSIFICATION

1) EVALUATION BASED ON TESTING DATA

Training and testing data are used separately for training and testing when we evaluate the effectiveness of the low-dimensional features extracted by our approach for two-category classification. Figure 3 shows the experimental results. Our STL-IDS performs better than single SVM. The accuracy, pre-cision, recall, and f-measure values for single SVM are 79.42%, 92.59%, 69.40%, and 79.42%, respectively. The cor-responding values for STL-IDS are 84.96%, 96.23%, 76.57%, and 85.28%, respectively. However, STL-IDS performs better in all performance metrics compared with single SVM. The experimental results also show that the proposed approach STL-IDS reduces training and testing times of SVM, as shown in Table 3.

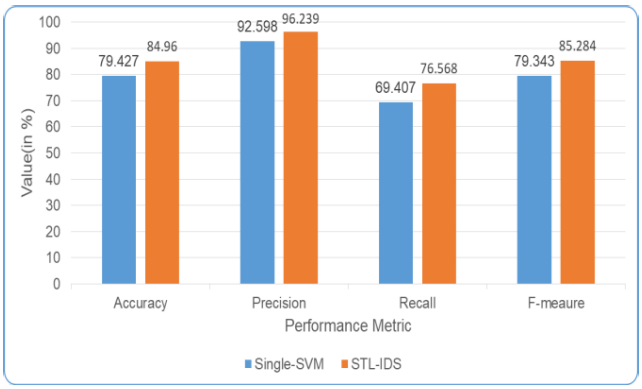

FIGURE 3. Accuracy, precision, recall, and F-measure values for STL-IDS and single SVM for binary classification based on test data.

2) EVALUATION BASED ON TRAINING DATA

In this section, we use 10-fold cross validation to evaluate the superiority of our proposed model through a comparison of its performance metrics and training and testing times with those of single SVM. Figure 4 shows that the performance of our STL-IDS in binary classification is higher than that of single SVM. The accuracy, precision, recall, and f-measure values are 99.416%, 99.45%, 99.291%, and 99.373%, respectively, whereas single SVM achieves 99.35%, 98.98%, 99.62%, and 99.30%, respectively. STL-IDS performs better than single SVM in all performance metrics, except for recall. The recall values for STL-IDS and single SVM are 99.29% and 99.62%, respectively. Moreover, TABLE 4 shows that our approach can reduce the training and testing times (Table 4) of SVM, which is crucial for measuring the efficiency of network security applications.

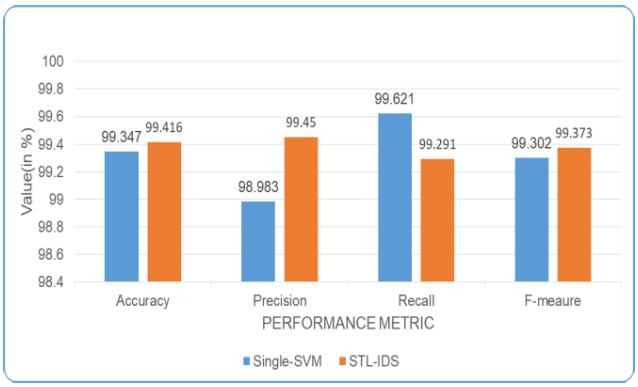

FIGURE 4. Accuracy, precision, recall, and F-measure values for STL-IDS and single SVM for binary classification based on training data.

B. EVALUATION THE IMPACT OF THE LOW-DIMENSIONAL FEATURES ON THE MULTICLASS CLASSIFICATION

1) EVALUATION BASED ON TESTING DATA

To verify the superiority of our proposed approach for network intrusion detection, we measure the performance of our model in five-category classification using testing data and compare it with that of single SVM in terms of performance metrics and training and testing times. Figure 5 shows a comparison between STL-IDS and single SVM. All the performance metric values of STL-IDS are higher than those of single SVM. The accuracy, precision, recall, and f-measure values for STL-IDS are 80.48%, 93.92%, 68.28%, and 79.078%, respectively, whereas those for single SVM are only 76.76%, 92.98%, 61.85%, and 74.28%, respectively. Moreover, Table 5 shows that the training and testing times of our proposed model are less than those of single SVM, indicating that our model is more concise and efficient than single SVM.

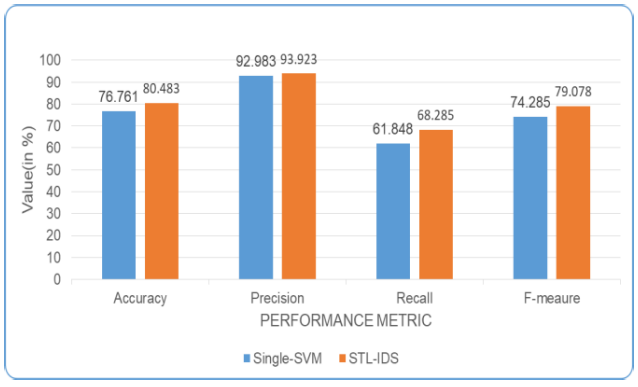

FIGURE 5. Accuracy, precision, recall, and F-measure values for STL-IDS and single SVM for five-category classification based on test data.

2) EVALUATION BASED ON TRAINING DATA

Similarly, we apply 10-fold cross-validation KDDTrain+ to evaluate the performance of our model for five-category classification. We also compare it with single SVM without feature learning and dimensionality reduction mechanisms of the raw dataset. Figure 6 provides a comparison of the performance metrics of our model and single SVM. The accuracy, precision, and f-measure values for STL-IDS are 99.396%, 99.56%, and 99.34%, respectively, whereas those for single SVM are 99.346%, 99.061%, and 99.288, respectively. STL-IDS has a lower recall value than single SVM. The recall values are 99.122% and 99.518% for STL-IDS and single SVM, respectively. However, STL-IDS for five-category classification is better than single SVM with regard to performance metrics and processing time. Table 6 shows that the training and testing times of STL-IDS are less than those of single SVM, indicating that our model is more efficient than single SVM in all situations.

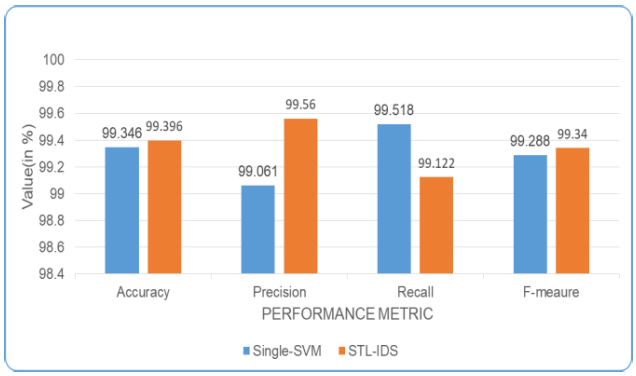

FIGURE 6. Accuracy, precision, recall, and F-measure values for STL-IDS and single SVM for five-category classification based on training data.

TABLE 6. Training and testing time and number of support vectors (NSV) comparison for STL-IDS and single SVM for five-category classification based on training data.

Method	Training Time(sec)	Testing Time(sec)	nSV
Single SVM	48402.477	313.511	108917
STL-IDS	26119.197	49.563	72027

C. THE EFFECT OF HYPER-PARAMETERS AND HIDDEN UNITS NUMBER SETTING IN OUR MODEL EFFICIENCY

Based on the theoretical analysis of the SAE and STL, it shows that the hidden unit number and the sparse parameter are the main parameters influencing the classification accuracy and training time speed. Thus, the hyper-parameters optimization is a crucial challenge for developing and designing an effective deep learning model for network intrusion detection. In addition, how to investigate the effect of hidden units number and the sparse parameter on the performance of our model and decrease the training and testing time is another challenge.

We were able to increase the performance of the Single SVM with k-fold cross-validation strategy to search for the best parameters of the RBF kernel (C = 5.6569 and Gamma = 1.0667). Also, for the hyper-parameters tuning for

sparse autoencoder, we use cross-validation folds strategy on KDDTrain+ part of NSL-KDD dataset. After the best value of hyper-parameters is selected, our model is trained and tested again 10-cross validation on KDDTrain+ with the best values. Also, it was trained and tested with KDDTrain+ and KDDTest+, respectively. Optimization process of our model was performed over key hyper-parameters and their values are given in Table 7 and Table 8. In our experiments, our model gets a higher accuracy for binary classification, when $p = 0.50$, $\lambda = 0.000001$, $\beta = 3$, and epochs number $= 1000$. Our model gets a higher accuracy for multiclass classification, when p is 0.77, λ is 0.000005, β is 3, and epochs number is 500.

TABLE 7. The tested values of hyper-parameters for our model for five-category classification.

Hyper-parameter name	Values	The Best Value
p	0.04,0.05,0.1,0.2,0.3 ,0.4,0.5, ,0.6,0.7,0.77,0.8	0.50
λ	0.01,0.001,0.07,0.000001,0.000005 ,0.000007	0.000001
β	1,2,3,4,5,6	3
Fine-tuning epochs	100,300,500,700,800,1000	500

TABLE 8. The tested values of hyper-parameters for our model for binary classification.

Hyper-parameter name	Values	The Best Value
p	0.04,0.05,0.1,0.2,0.3 ,0.4,0.5, ,0.6,0.7,0.77,0.8	0.77
λ	0.01,0.001,0.07,0.000001,0.000005 ,0.000007	0.000005
β	1,2,3,4,5,6	3
Fine-tuning epochs	100,300,500,700,800,1000	1000

In other experiments, we show how we investigate the sparse autoencoder ability for feature learning and dimensionality reduction of our data for enhancing the accuracy and decreasing the training and testing times of SVM. Thus, we restrict the number of hidden layer units to be less than the original input units and we can get a compressed representation, which actually achieves the desired dimensionality reduction effect. In our experiments, our model gets a higher accuracy, when the number of hidden units is 30 and epochs number is 1000. Less training and testing times of SVM compared with Single SVM, which we discussed in above subsections, almost when the number of hidden units are less than half of the number of the original input units (less of 60 hidden units). Figure 7, Figure 8, and Figure 9 show the test classification accuracy of our model, training and testing times of SVM with different numbers of hidden units.

Finally, to test the effect of the sparse parameter on the classification accuracy and training time speed. We use KDDTrain+ as our train data and KDDTest+ for testing for

FIGURE 7. The accuracy and testing time on KDDTEST+ dataset in the binary classification with different number of hidden units.

FIGURE 8. The accuracy and testing time on KDDTEST+ dataset in the multiclass classification with different number of hidden units.

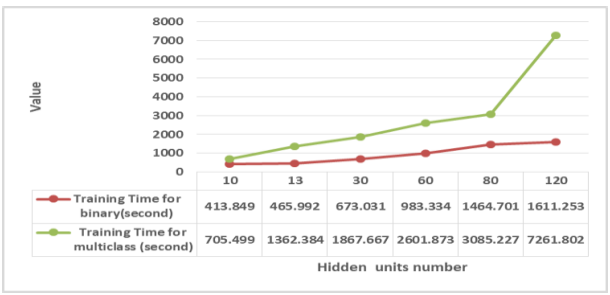

FIGURE 9. The training time on KDDTRAIN+ dataset in the binary and multiclass classification with different number of hidden units.

our experiments. We use the best values in TABLE 7 and TABLE 8 to other hyper-parameters in our model. The number of hidden units are 30 and 13 for binary and multiclass classification, respectively. As shown in Figure 10 and

Figure 11, the optimal classification accuracy is obtained when the sparse parameter equals to 0.77 and 0.50 for the binary and multiclass classification, respectively.

In this section, we explored how to exploit the impact of the sparse parameter and the hidden unit number on our model to obtain a higher classification accuracy and a lower training times.

D. DISCUSSION AND ADDITIONAL COMPARISONS

We also verify our model's superiority by comparing its detection accuracy with that obtained from other classification algorithms in related studies. Yin *et al.* [22] claimed that their model, which was constructed with recurrent neural network and soft-max classifier and applied on KDDTest+ for

TABLE 9. Additional performance comparisons with several related approaches in the binary classification.

Method	Accuracy Rate	Training Datasets	TESTING DATASETS	EVALUATION TYPE
J48 [35]	81.07%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Naïve Baysian [35]	75.56%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Random forest [35]	80.67%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
NB-Tree [35]	82.02%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Multi-layer Perceptron [35]	77.41%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
augmented features+ SVM [23]	99.31%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTrain+)	evaluation based on KDDTrain+ using 10-fold cross-validation
fuzziness approach[21]	84.12%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Recurrent Neural Network [22]	83.28%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
AE+ Gaussian naïve Bayes [10]	83.34%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
SVM+ELM[40]	95.75%	10% KDD 99 (494,021 records)	KDD99 Corrected (311,029 records)	evaluation based on KDD99 Corrected
DAE-IDS[27]	96.53%	10% KDD 99 (145,586 records)	KDD99 Corrected (77291 records)	evaluation based on KDD99 Corrected
SAE+SMR[2]	88.39%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
SAE+SMR [2]	< 99%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTrain+)	evaluation based on KDDTrain+ using 10-fold cross-validation
Our Model	95.09%	10% KDD 99 (145,586 records)	KDD99 Corrected (77291 records)	evaluation based on KDD99 Corrected
Our Model	99.416%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTrain+)	evaluation based on KDDTrain+ using 10-fold cross validation
Our Model	84.96%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +

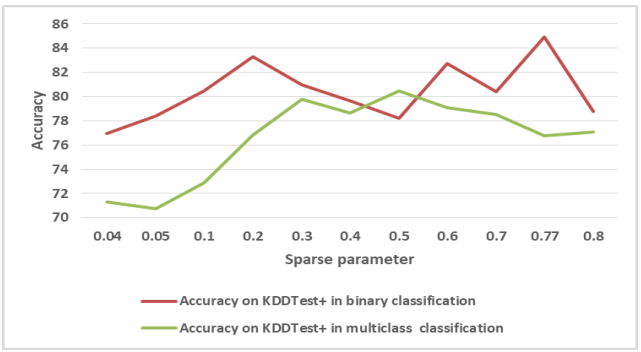

FIGURE 10. Accuracy on KDDTEST+ dataset in the binary and multiclass classification with different sparse parameer.

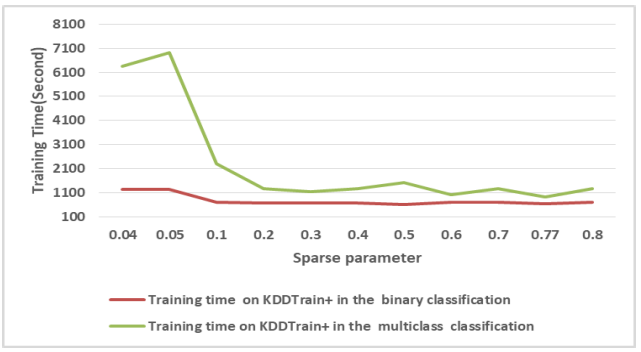

FIGURE 11. The training time on KDDTRAIN+ dataset in the binary and multiclass classification with different sparse parameter.

evaluation, obtained 83.28% and 81.29% detection accuracy for two-category and five-category classification, respectively. The authors compared the results of their model with those of many classification algorithms discussed in [22] and [35], as demonstrated in Table 9 and Table 10. Table 9 and Table 10 show that our model is a competitor of [22] in terms of accuracy in two-category classification since we improved SVM with a suitable learning algorithm (SAE) in the STL approach. Moreover, our model is a competitor in terms of time complexity since we used the SAE technique for dimensionality reduction and data representation.

TABLE 10. Additional performance comparisons with several related approaches in the multiclass classification.

Method	Accuracy Rate	Training Datasets	Testing Datasets	evaluation Type
J48 [22]	74.60%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Naïve Baysian [22]	74.40	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Random forest [22]	72.80	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
NB-Tree [22]	75.40%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Multi-layer Perceptron [22]	78.10%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
Recurrent Neural Network[22]	81.29%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
AutoEncoder+DBN$^{10\text{-}1}$0[41]	95.75%	10% KDD 99 (494,021 records)	KDD99 Corrected (311,029 records)	evaluation based on KDD99 Corrected
DBN4[42]	93.49%	10% KDD 99 (494,021 records)	KDD99 Corrected (11850 records)	evaluation based on KDD99 Corrected
DBN^{4}+LR[43]	97.90%	10% KDD 99 (494,021 records)	KDD99 Corrected (11850 records)	evaluation based on KDD99 Corrected
LSTM[44]	93.82%	10% KDD 99 (494,021 records)	KDD99 Corrected (300000 records)	evaluation based on KDD99 Corrected
DAE-IDS[27]	94.71%	10% KDD 99 (145,586 records)	KDD99 Corrected (77291 records)	evaluation based on KDD99 Corrected
SAE+SMR [2]	79.10%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTest+)	evaluation based on KDDTest +
SAE+SMR [2]	< 99%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTrain+)	evaluation based on KDDTrain+ using 10-fold cross-validation
Our Model	99.396%	NSL-KDD (KDDTrain+)	NSL-KDD (KDDTrain+)	evaluation based on KDDTrain+ using 10-fold cross validation
Our Model	80.48%	NSL-KDD (KDDTrain+)	NSL-KDD(KDDTest+)	evaluation based on KDDTest +
Our Model	93.960%	10% KDD 99 (145,586 records)	KDD99 Corrected (77291 records)	evaluation based on KDD99 Corrected

Wang *et al.* [23] also claimed that their model achieved an accuracy of 99.31% for binary classification on the basis of 10-fold cross validation for KDDTrain+ (training data part of the NSL-KDD dataset).

Table 9 demonstrates that our model achieves better accuracy compared with [23] model. However, their model performs better in terms of training time. Wang *et al.* [23] used logarithm marginal density ratio transformation (LMDRT) to verify their model's efficiency in enhancing the accuracy and reducing the training time of SVM. The weakness of their method is that the reduction in testing time is not considered, unlike in our proposed method. Yousefi-Azar *et al.* [10] evaluated their feature learning model based on deep AE, and they applied their model with many shallow machine learning algorithms, such as SVM.

However, the highest accuracy they achieved was 83.30%. As demonstrated in Table 9, our model is superior to their model. Moreover, their model is computationally expensive because it contains many hidden layers and involves two training stages.

Javaid *et al.* [2] claimed that their deep learning approach for network intrusion detection, which depends on STL that combines SAE with soft-max, obtains an accuracy of 88.39% and 79.10% for two-category and five-category classification, respectively. Their approach was evaluated on KDDTest+ (the testing data of NSL-KDD dataset). Their model obtained an accuracy of less than 99% for two-category and five-category classification based on KDDTrain+, which was evaluated using 10-fold cross-validation. The experimental results show that our method outperforms the model in [2] by 1.38% in terms of detection accuracy rate when applied on KDDTrain+ and KDDTest+ separately for training and testing for five-category classification. For the KDDTrain+ dataset, our method achieves better results than [2]. Its accuracy for two-category and five-category classification is 99.423% and 99.414%, respectively. This experimental evidence and the comparison of our method and with that of [2] demonstrate that our method achieves better results than [2] in terms of detection accuracy rate and time complexity because we used good feature

learning algorithm (SAE) with strong classifier (SVM) instead of weak classifier (soft-max). We used SAE not only for feature learning that produces a new good representation of our dataset that leads to a good performance but also for the dimensionality reduction that leads to obtain the low-dimensional features that improve SVM classification accuracy and decrease the training and testing times of the algorithm. In order to compare the performance of STL-IDS with some recent related works which evaluated with KDD-CUP'99 dataset, we evaluated our proposed model on the KDD-CUP'99 dataset. We used The 10%KDDCup dataset contains 494,021 samples for training phase and corrected labels KDDCup 99 dataset contains 311029 samples for testing phase. After we removed all duplicate samples in both dataset to avoid the bias towards more frequent samples, the training and testing datasets consist of 145,586 and 77,291 instances, respectively. The results of this experiment show that our model has achieved higher accuracy rate than most of the existing methods on the same dataset.

The accuracy of our model is lower than that of DAE-IDS [27], and the gap remains within 0.7% and 1.4% for multiclass and binary classification, respectively. This indicates that our model has reached or exceeded the average overall accuracy level of other state-of-the-art approaches and methods. Moreover, Table 9 and Table 10 demonstrate that the performance of our proposed method is very close to or more than other state-of-the-art approaches in terms of accuracy rate.

V. CONCLUSION AND FUTURE WORK

The proposed approach is another means of utilizing the STL framework based on SAE for feature learning and dimensionality reduction and using SVM instead of soft-max

for classification. The experimental results of the proposed approach show that our model demonstrates improved SVM classification accuracy and accelerated training and testing times. It also exhibits good performance in two-category and five-category classification. Compared with other previous models and shallow classification methods, such as J48, naive Bayesian, RF, and SVM, our approach achieved a higher accuracy rate particularly under five-category classification, on the NSL-KDD dataset. The future expansion of our proposed approach will focus on further improvement by using multiple stages of STL and a hybrid feature learning model for good representation features and dimensionality reduction mechanisms. Additionally, the model's training and

testing times can be further reduced by the implementation of the system in parallel platforms or GPU acceleration.

ACKNOWLEDGEMENTS

The authors would like to thank the anonymous referees for their helpful comments and suggestions.

CONFLICTS OF INTEREST

The authors declare no conflict of interest.

REFERENCES

[1] M. Ahmed, A. N. Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *J. Netw. Comput. Appl.*, vol. 60, pp. 19–31, Jan. 2016.

[2] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun. Technol. (Formerly BIONETICS)*, 2016, pp. 21–26.

[3] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 1, pp. 41–50, Feb. 2018.

[4] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *Proc. 8th IEEE Int. Conf. Commun. Softw. Netw. (ICCSN)*, Jun. 2016, pp. 581–585.

[5] L.-S. Chen and J.-S. Syu, "Feature extraction based approaches for improving the performance of intrusion detection systems," in *Proc. Int. MultiConf. Eng. Comput. Scientists*, vol. 1, Mar. 2015, pp. 1–6.

[6] S. Seo, S. Park, and J. Kim, "Improvement of network intrusion detection accuracy by using restricted Boltzmann machine," in *Proc. 8th Int. Conf. Comput. Intell. Commun. Netw. (CICN)*, Dec. 2016, pp. 413–417.

[7] R. Salakhutdinov and H. Larochelle, "Efficient learning of deep Boltzmann machines," in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, 2010, pp. 693–700.

[8] M. Z. Alom, V. R. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *Proc. Nat. Aerosp. Electron. Conf. (NAECON)*, Jun. 2015, pp. 339–344.

[9] M. A. Salama, H. F. Eid, R. A. Ramadan, A. Darwish, and A. E. Hassanien, "Hybrid intelligent intrusion detection scheme BT," in *Soft Computing in Industrial Applications*. Berlin, Germany: Springer, 2011, pp. 293–303.

[10] M. Yousefi-Azar, V. Varadharajan, L. Hamey, and U. Tupakula, "Autoencoder-based feature learning for cyber security applications," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 3854–3861.

[11] G. Kim, S. Lee, and S. Kim, "A novel hybrid intrusion detection method integrating anomaly detection with misuse detection," *Expert Syst. Appl.*, vol. 41, no. 4, pp. 1690–1700, 2014.

[12] Y. Liao and V. R. Vemuri, "Use of *K*-nearest neighbor classifier for intrusion detection," *Comput. Secur.*, vol. 21, no. 5, pp. 439–448, 2002.

[13] L. Koc, T. A. Mazzuchi, and S. Sarkani, "A network intrusion detection system based on a hidden Naïve Bayes multiclass classifier," *Expert Syst. Appl.*, vol. 39, no. 18, pp. 13492–13500, 2012.

[14] S. Mukherjee and N. Sharma, "Intrusion detection using naive bayes classifier with feature reduction," *Procedia Technol.*, vol. 4, pp. 119–128, Feb. 2012.

[15] E. de la Hoz, E. de la Hoz, A. Ortiz, J. Ortega, and A. Martínez-Álvarez, "Feature selection by multi-objective optimisation: Application to network anomaly detection by hierarchical self-organising maps," *Knowl.-Based Syst.*, vol. 71, pp. 322–338, Nov. 2014.

[16] E. De la Hoz, E. De La Hoz, A. Ortiz, J. Ortega, and B. Prieto, "PCA filtering and probabilistic SOM for network intrusion detection," *Neurocomputing*, vol. 164, pp. 71–81, Sep. 2015.

[17] R. Sen, M. Chattopadhyay, and N. Sen, "An efficient approach to develop an intrusion detection system based on multi layer backpropagation neural network algorithm: IDS using BPNN algorithm," in *Proc. ACM SIGMIS Conf. Comput. People Res.*, 2015, pp. 105–108.

[18] T. Mehmood and H. B. M. Rais, "SVM for network anomaly detection using ACO feature subset," in *Proc. Int. Symp. Math. Sci. Comput. Res. (iSMSC)*, May 2015, pp. 121–126.

[19] S. Mukkamala, G. Janoski, and A. Sung, "Intrusion detection using neural networks and support vector machines," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 2, May 2002, pp. 1702–1707.

[20] G. Kou, Y. Peng, Z. Chen, and Y. Shi, "Multiple criteria mathematical programming for multi-class classification and application in network intrusion detection," *Inf. Sci.*, vol. 179, no. 4, pp. 371–381, 2009.

[21] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Inf. Sci.*, vol. 378, pp. 484–497, Feb. 2017.

[22] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[23] H. Wang, J. Gu, and S. Wang, "An effective intrusion detection framework based on SVM with feature augmentation," *Knowl.-Based Syst.*, vol. 136, pp. 130–139, Nov. 2017.

[24] D. Perez, M. A. Astor, D. P. Abreu, and E. Scalise, "Intrusion detection in computer networks using hybrid machine learning techniques," in *Proc. 43rd Latin Amer. Comput. Conf. (CLEI)*, Sep. 2017, pp. 1–10.
[25] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for Internet of Things," *Future Gener. Comput. Syst.*, vol. 82, pp. 761–768, May 2018.
[26] W. Wang *et al.*, "HAST-IDS: Learning hierarchical spatial-temporal features using deep neural networks to improve intrusion detection," *IEEE Access*, vol. 6, pp. 1792–1806, 2018.
[27] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *Proc. 20th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2018, p. 1.
[28] P. Madani and N. Vlajic, "Robustness of deep autoencoder in intrusion detection under adversarial contamination," in *Proc. 5th Annu. Symp. Bootcamp Hot Topics Sci. Secur.*, 2018, Art. no. 1.
[29] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. 24th Int. Conf. Mach. Learn.*, 2007, pp. 759–766.
[30] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, 2011, pp. 215–223.
[31] A. Ng, "Sparse autoencoder. CS294A lecture notes," Stanford Univ., Stanford, CA, USA, Tech. Rep. 72, 2011.
[32] H. Liu, T. Taniguchi, T. Takano, Y. Tanaka, K. Takenaka, and T. Bando, "Visualization of driving behavior using deep sparse autoencoder," in *Proc. IEEE Intell. Vehicles Symp.*, Jun. 2014, pp. 1427–1434.
[33] J. Deng, Z. Zhang, E. Marchi, and B. Schuller, "Sparse autoencoder-based feature transfer learning for speech emotion recognition," in *Proc. Humaine Assoc. Conf. Affect. Comput. Intell. Interact.*, Sep. 2013, pp. 511–516.
[34] C.-F. Tsai, Y.-F. Hsu, C.-Y. Lin, and W.-Y. Lin, "Intrusion detection by machine learning: A review," *Expert Syst. Appl.*, vol. 36, no. 10, pp. 11994–12000, 2009.
[35] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *Proc. IEEE Symp. Comput. Intell. Secur. Defense Appl.*, Jul. 2009, pp. 1–6.
[36] N. Paulauskas and J. Auskalnis, "Analysis of data pre-processing influence on intrusion detection using NSL-KDD dataset," in *Proc. Open Conf. Elect. Electron. Inf. Sci. (eStream)*, Apr. 2017, pp. 1–5.
[37] C. L. C. Chang. *Libsvm*. Accessed: Jan. 5, 2018. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/libsvm/
[38] S. Maji, A. C. Berg, and J. Malik, "Classification using intersection kernel support vector machines is efficient," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2008, pp. 1–8.
[39] P. Ilayaraja, N. V. Neeba, and C. V. Jawahar, "Efficient implementation of SVM for large class problems," in *Proc. 19th Int. Conf. Pattern Recognit.*, Dec. 2008, pp. 1–4.
[40] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified k-means for intrusion detection system," *Expert Syst. Appl.*, vol. 67, pp. 296–303, Jan. 2017.
[41] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," *Methods*, vol. 9, no. 5, pp. 205–216, 2015.
[42] N. Gao, L. Gao, Q. Gao, and H. Wang, "An intrusion detection model based on deep belief networks," in *Proc. 2nd Int. Conf. Adv. Cloud Big Data*, Nov. 2014, pp. 247–252.
[43] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. 15th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Anaheim, CA, USA, Feb. 2017, pp. 195–200.
[44] R. C. Staudemeyer, "Applying long short-term memory recurrent neural networks to intrusion detection," *South Afr. Comput. J.*, vol. 56, no. 1, pp. 136–154, 2015.

MAJJED AL-QATF received the B.S. degree in network technology and computer security from Sana'a University, Sana'a, Yemen, in 2013. He is currently pursuing the M.S. degree in computer science with the School of Information Science and Engineering, Central South University, Changsha, China. His research interests include deep learning, information security, and data mining.

YU LASHENG received the B.Sc. degree in computer science and the M.S. and Ph.D. degrees in control theory and control engineering from Central South University, China. He is currently a Vice Professor with Central South University. He has authored at least 70 papers on agent technologies or algorithms and three books. He has organized and implemented many projects that have greatly benefitted our society. His main research interests include smart computing, agent technologies and applications, structure and algorithm, and distributed computing. He is an ACM and CCF Member, and an ACM/ICPC Golden Medal Coach. He is an Editor of the *Journal of Convergence Information Technology* and the *Advances in Information Sciences and Service Sciences*. He is also a reviewer for *Future Generation Computer Systems, Journal of Parallel and Distributed Computing, Artificial Intelligence Review*, and some other journals and conferences.

MOHAMMED AL-HABIB received the B.S. degree in computer sciences and information systems from Thamar University, Thamar, Yemen, in 2011. He is currently pursuing the M.S. degree in computer science with the School of Information Science and Engineering, Central South University, Changsha, China. His research interests include deep learning, computer vision, and data mining.

KAMAL AL-SABAHI received the B.S. degree in computer science from Sana'a University, Sana'a, Yemen, in 2008, and the M.S. degree in information technology from OUM University, Kuala Lumpur, Malaysia, in 2015. He is currently pursuing the Ph.D. degree in computer science with the School of Information Science and Engineering, Central South University, Changsha, China. His research interests include deep learning, natural language processing, knowledge engineering, and data mining.