Received January 1, 2019, accepted January 23, 2019, date of publication February 15, 2019, date of current version March 20, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2899721

A Novel Two-Stage Deep Learning Model for Efficient Network Intrusion Detection

FARRUKH ASLAM KHAN[1], (Senior Member, IEEE), ABDU GUMAEI[2], ABDELOUAHID DERHAB[1], AND AMIR HUSSAIN[3], (Senior Member, IEEE)

[1]Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia
[2]Department of Computer Science, King Saud University, Riyadh, Saudi Arabia
[3]School of Computing, Edinburgh Napier University, Edinburgh EH10 5DT, Scotland, U.K.

Corresponding author: Farrukh Aslam Khan (fakhan@ksu.edu.sa)

This work was supported by the Deanship of Scientific Research at King Saud University, Saudi Arabia, through the Research Group under Project RGP-214. The work of A. Hussain was in part supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) under Grant EP/M026981/1 (AV-COGHEAR).

ABSTRACT The network intrusion detection system is an important tool for protecting computer networks against threats and malicious attacks. Many techniques have recently been proposed; however, these face significant challenges due to the continuous emergence of new threats that are not recognized by existing systems. In this paper, we propose a novel two-stage deep learning (TSDL) model, based on a stacked auto-encoder with a soft-max classifier, for efficient network intrusion detection. The model comprises two decision stages: an initial stage responsible for classifying network traffic as normal or abnormal, using a probability score value. This is then used in the final decision stage as an additional feature, for detecting the normal state and other classes of attacks. The proposed model is able to learn useful feature representations from large amounts of unlabeled data and classifies them automatically and efficiently. To evaluate its effectiveness, several experiments are conducted on two public datasets, specifically the benchmark KDD99 and UNSW-NB15 datasets. Comparative simulation results demonstrate that our proposed model significantly outperforms existing approaches, achieving high recognition rates, up to 99.996% and 89.134%, for the KDD99 and UNSW-NB15 datasets respectively. We conclude that our model has the potential to serve as a future benchmark for the deep learning and network security research communities.

INDEX TERMS Computational intelligence, two-stage deep learning model, feature representation, network intrusion detection, stacked auto-encoder.

I. INTRODUCTION

With the growing use of computer networks across different fields and applications, network security is becoming increasingly important. Many organizations use traditional security tools such as firewalls, anti-spam techniques, antiviruses, etc., to protect against network attacks. Unfortunately, these are unable to recognize new and sophisticated attacks. Recently, Network Intrusion Detection Systems (NIDSs) have emerged as a second line of defence to monitor network activity and detect intrusive events. They are now considered as powerful defence tools that can protect against sophisticated attacks and threats [1].

Network intrusion detection is not a trivial task [2], [3]. There are several problems and challenges faced by network intrusion detection approaches for efficiently and effectively recognizing anomalies [4]. The first problem arises from the variety and variability of malicious attacks and threats. Existing intrusion detection methods are unable to cope with continuous evolutions in the cyber threat landscape and the emergence of new threats; hence many are inefficient at achieving high detection rates or reducing false alarms. The second problem is that classical machine learning methods used in network intrusion detection approaches present several difficulties, such as overfitting and high bias due to irrelevant or redundant features, and the imbalanced class distribution of network traffic [5]. The third problem is related to the complexity of labeling the traffic dataset for developing a NIDS [6]. Substantial efforts are required to produce such

The associate editor coordinating the review of this manuscript and approving it for publication was Yin Zhang.

labeled datasets over a period of time. These problems make intrusion detection methods ineffective at detecting real-world threats in large-scale environments. Moreover, they are unable to efficiently learn feature representations in order to build a more effective predictive model.

Recently, deep-learning-based methods have been successfully applied in, amongst other tasks, image recognition, speech recognition [7], and action recognition [8], [9], etc. These methods aim to learn relevant features from a large sample of unlabeled data features and subsequently apply them to a limited amount of labeled data features, in a supervised learning fashion. The labeled and unlabeled data may come from different distributions, but they must be relevant to each other [10]. For example, Ahsan *et al.* [11] developed a novel Random Neural Network (RNN) for real-time cognitive system design requirements. They integrated the RNN with a genetic algorithm to achieve real-time decision-making. Other recent approaches have proposed alternative multi-layered echo state network, deep, reinforcement, and multi-task learning methods [12]–[15].

Thus far, few works have utilized deep learning algorithms for network intrusion detection, for instance, deep belief networks (DBNs), restricted Boltzmann machines (RBMs), stacked auto-encoders (SAEs), and supervised learning with convolutional neural networks (CNNs). Even though CNNs can reduce the number of parameters through strategies of sparse connectivity and shared weights, these algorithms are designed for supervised learning and require a large amount of labeled network data as input, which is a rather costly option.

In the past, before the emergence of deep learning concepts, the weights of a supervised neural network with two or more hidden layers in the training phase were randomly initialized from a Gaussian distribution. Subsequently, a back-propagation optimization method was applied in order to find the optimal parameters. In practice, this method based on random initialization was shown to lead to slow optimization and poor local minima solutions, since the loss function is extremely uncontrolled when parameterized by millions of correlated variables. To accommodate these limitations, Hinton *et al.* [16] proposed the concept of deep learning to optimize the learning problem, by pre-training each layer of the network in an unsupervised way, to learn a discriminative representation of the data before the classification task. Additionally, Vincent *et al.* [17] proposed a method, based on a stacked denoising auto-encoder (SDAE) to enable a deep neural network to learn a useful representation of features. With use of effective feature learning, redundant or irrelevant features in the training data, which can lead to overfitting, are reduced. A key reason for the overfitting problem in a real-time NIDS is feature redundancy in the training data of the system model.

In this paper, we exploit the deep stacked auto-encoder (DSAE) as part of a novel cascade architecture, termed two-stage deep learning (TSDL), for feature learning and dimensionality reduction, in which the output of the first stage is used as input in the next stage, with optimised feature representation and redundancy reduction. The proposed model differs from previous works as it learns feature representation in two stages. The first stage learns feature representation for classifying normal and abnormal network traffic with a probability score value. This value is used as an additional feature in the second stage in order to learn feature representation of normal traffic and other types of attacks. The first stage aims to avoid the overfitting problem and to mitigate the bias towards normal traffic, through an increased focus on abnormal traffic. This not only helps to confirm the detection of the normal class but also contributes to the classification of other types of attacks. The purpose of this research is to build a two-stage feature-learning model based on deep learning for improving accuracy and reducing false alarm rates of the NIDS.

The main contributions of our work can be summarized as follows:

1) We propose a novel two-stage semi-supervised feature-learning model for network intrusion detection, with two modes of operations, i.e., two-class and multiclass intrusion detection. The proposed model has a unique topology and architecture compared to existing works in this domain.
2) We introduce a low-cost DSAE approach for NIDSs, which uses a reduced number of features compared to other state-of-the-art approaches (5 abstract features for 2-class intrusion detection and 10 abstract features for multiclass intrusion detection). This makes the proposed model more efficient for real-time detection.
3) We optimize the parameters of the proposed model using best practices reported in deep learning research [18].
4) We demonstrate the model's effectiveness in detecting sophisticated new attacks by conducting experiments on two benchmark public datasets, namely, an older KDD99 dataset and the new UNSW-NB15 dataset.

The remainder of this paper is organized as follows: in section II, we present related work on machine learning and deep-learning-based intrusion detection models and approaches. In section III, we provide background knowledge on general auto-encoder neural networks. In section IV, we introduce the methodology used in this study in more detail. In section V, we describe the experiments and evaluation, including tools and datasets used, the data pre-processing step, and comparative experimental results. Finally, in section VI, we summarize our conclusions and outline future work directions.

II. RELATED WORK

Over the past few years, a number of models and approaches based on traditional machine learning have been proposed for network intrusion detection. Examples include the Support Vector Machine (SVM) [24], [25], K-Nearest Neighbors (KNN) [26], Artificial Neural Networks

(ANN) [17], [18], Random Forests (RF) [24], [25], [59], Decision Trees (DT) [23], [26], [27], [60], [61], Linear Regression (LR), Naïve Bayes (NB), Expectation Maximization (EM) [23], Simulated Annealing (SA) [27], Simplified Swarm Optimization (SSO) [28], Neutrosophic Logic (NL) [29], Neurotree [30], Random Effects Logistic Regression (RELR) [31], PCA filtering [62], and others reported in [32]–[34]. Recently, Nawir *et al.* [35] proposed a classification model based on the Average One Dependence Estimator (AODE) algorithm for multiclass classification, on the UNSW-NB15 dataset, reporting an accuracy of 83.47% and a false alarm rate (FAR) of 6.57%. Janarthanan and Zargari [36] applied the RF classifier on the UNSW-NB15 dataset to detect and classify intrusion attacks with five selected features, with an accuracy of 81.6175% and FAR of 4.4%.

Deep learning, a branch of machine learning, has recently been used for network intrusion detection. State-of-the-art deep learning approaches and methods that have been used for unsupervised feature learning for network intrusion detection, include deep belief networks (DBNs), deep neural networks (DNNs), restricted Boltzmann machines (RBMs), auto-encoders, and variations of these methods. For example, Erfani *et al.* [37] proposed a new approach, which combined DBNs with a linear one-class SVM for intrusion detection and applied it on several benchmark datasets. Similarly, Fiore *et al.* [38] presented a discriminative RBM (DRBM) method to learn compressed features from specific features that are not in the packet payloads. The compressed features are fed into a soft-max classifier to create a binary classification of normal and abnormal behaviors. Javaid *et al.* [39] presented a deep learning method based on DNNs for anomaly detection. Their results showed that deep learning is more effective for flow-based anomaly detection in software-defined networks (SDNs). Tang *et al.* [40] introduced a deep learning approach for network intrusion detection based on self-taught learning (STL), applied to the NSL-KDD dataset. Their experimental results showed that deep learning surpasses previous studies in terms of performance and accuracy. Wang [41] proposed a deep learning approach based on the stacked auto-encoder for network traffic recognition from raw data and attained remarkably high performance. Additionally, Yin *et al.* [42] proposed a deep learning method built on recurrent neural networks (RNNs) for intrusion detection. This method was applied on the NSL-KDD dataset and demonstrated that deep learning methods for intrusion detection outperform traditional machine learning classification approaches. Alrawashdeh and Purdy [43] also introduced a deep learning approach based on a DBN of RBMs with one and four hidden layers for feature reduction. The weights of the DBN were updated in a fine-tuning phase and the classification task was performed using a Logistic Regression classifier. The proposed approach was implemented on the KDD99 dataset and achieved an accuracy of 97.9% with a false alarm rate of 0.5%. Nevertheless, the accuracy on this old benchmark dataset is still not high enough to make this a strong approach for network intrusion detection.

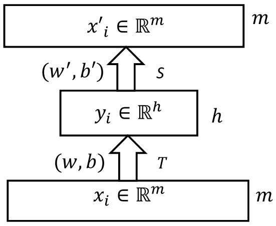


FIGURE 1. An $m/h/m$ auto-encoder architecture.

Shone *et al.* [44] presented a deep learning approach for intrusion detection based on a non-symmetric deep auto-encoder (NDAE). This technique was applied on the KDD99 dataset using RF as a classifier and achieved an accuracy rate up to 97.85%. However, the false alarm rate was as high as 2.15%, which makes this method ineffective for the detection of sophisticated attacks. More recently, Nguyen *et al.* [45] proposed a framework based on principal component analysis (PCA) and a Gaussian-binary restricted Boltzmann machine (GRBM) to detect cyber attacks in the mobile cloud environment. However, the testing methodology of this approach is unclear and not sufficiently detailed to enable comparative benchmarking.

In order to recognize more sophisticated attacks, the model proposed in this paper is designed to learn more useful feature representations from large amounts of unlabeled features. Moreover, to improve the flexibility of current NIDSs, the proposed model can detect network intrusions in two modes of operations, namely normal and abnormal state detection, in addition to detecting normal states and other types of attacks. Unlike existing models, we propose a two-stage deep learning (TSDL) model for network intrusion detection: a cascade model where the probability score value of normal and abnormal state classification resulting from the first stage is used as an additional feature for the second stage, in order to classify the normal state and different types of attacks.

III. PRELIMINARIES OF THE GENERAL AUTO-ENCODER

In this section, we briefly introduce the general notion of an auto-encoder for the purpose of data reduction. Fundamentally, an $m/h/m$ auto-encoder is mathematically defined by a tuple $m, h, n, \mathbb{R}, \mathcal{T}, \mathcal{S}, \mathcal{X}, Y, X', \mathcal{D}$, as shown in Figure 1:

- $\mathbb{R}$ is a set of real numbers.
- m and h are positive integers that represent the length of X and Z. For data reduction, the case $0 < h < m$ is considered.
- $\mathcal{T}$ is a mapping function from $\mathbb{R}^m$ to $\mathbb{R}^h$.
- $\mathcal{S}$ is a mapping function from $\mathbb{R}^h$ to $\mathbb{R}^m$.
- $X = \{x_1, x_2, \ldots, x_n\}$ is a set of n training vectors in $\mathbb{R}^m$. When external targets exist, $Y = \{y_1, y_2, \ldots, y_n\}$ is a set of n compressed vectors in $\mathbb{R}^h$, and $X' = \{x_1', x_2', \ldots, x'_n\}$ represents the equivalent set of target vectors in $\mathbb{R}^m$.

- $\mathcal{D}$ is a distortion or dissimilarity function (e.g., the *Lp* norm or Hamming distance), defined over $\mathbb{R}^m$.

For any transformations $T \in \mathcal{T}$ and $S \in \mathcal{S}$, the auto-encoder transforms the input training vectors $x_i \in \mathbb{R}^m$ into output vectors $T \circ S(x_i) \in \mathbb{R}^m$. The auto-encoder learns the weights and biases of the transformations T and S using the training vectors by minimizing the distortion function as defined in the following equation:

$$E(T, S) = \sum_{i=1}^{m} E(x_i) = \sum_{i=1}^{m} \mathcal{D}(S(T(x_i; w, b); w', b'), x_i) \quad (1)$$

Where the weight matrix w and the bias vector b are the parameters of T, and the weight matrix w' and the bias vector b' are the parameters of S. When $h < m$, the auto-encoder projects the data on a lower dimensional space, and thus performs the data reduction and compression.

By using the general architecture of the auto-encoder, different kinds of feature representations can be obtained, based on the best choice of mappings $\mathcal{T}$ and $\mathcal{S}$, the distortion function $\mathcal{D}$, and the use of additional constraints such as generalization and regularization. A deep neural network is defined by stacking a number of auto-encoders. For many classification problems, non-linear sigmoid activation is usually applied on the hidden layers. In the case of mapping from $\mathbb{R}^m$ to $\mathbb{R}^h$, the auto-encoders can work as though corresponding to T and S, which are classes of linear transformation functions. Therefore, T and S are matrices of size $h \times m$ and $m \times h$, respectively. The linear transformation of $\mathbb{R}^m$ and $\mathbb{R}^h$ using $\mathcal{D}$, which is a squared Euclidean distance, was presented in [46], [47]. Moreover, the theory of complex-valued linear auto-encoders has also been extended in [48].

IV. METHODOLOGY

In this section, we first present the theoretical concept behind our proposed model. Then, the proposed TSDL model for network intrusion detection is described in more detail. Following this, we briefly introduce the two public datasets used to evaluate the model. Finally, we explain our method for pre-processing the two datasets before describing the simulation experiments and comparisons with state-of-the-art approaches.

A. THEORETICAL CONCEPTS BEHIND THE PROPOSED MODEL

The basic unit of the proposed TSDL model is the auto-encoder (AE). An AE is essentially a feed-forward neural network similar to a multi-layer perceptron (MLP), composed of three main layers, namely the input layer, one or more hidden layer(s) and the output layer, where the numbers of neurons in the output and input layers are equal. Figure 2 shows a typical architecture of a simple AE consisting of a single hidden layer.

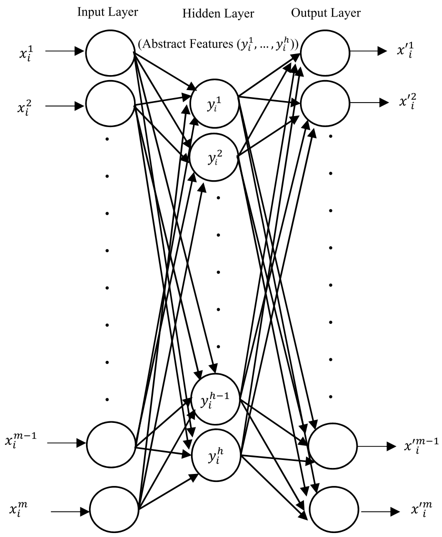


FIGURE 2. The basic architecture of the AE with a single hidden layer.

The learning technique of an AE is unsupervised learning because the AE learns the abstract compressed data representation for reconstructing the original input data instead of guessing the target output from the input. An AE includes two processes: the encoding process, which takes place between the input layer and the hidden layer, and the decoding process, which takes place between the hidden layer and the output layer.

Suppose that the input layer of AE contains m nodes for n input data vectors x_i and h nodes in the hidden layer for the equivalent set of abstract compressed data vectors y_i. The number of nodes in the output layer is equal to the number of nodes in the input layer and the reconstructed vectors x_i' are approximately equal to the input vectors, which are expressed as follows:

$$x_i \cong x'_i, \quad \text{for } i = 1, 2, \dots, n \quad (2)$$

In the encoding process of the AE, the abstract compressed data vectors y_i can be computed as follows:

$$T : y_i = g(x_i w + b), \quad \text{for } i = 1, 2, \dots, n \quad (3)$$

where w, b and x_i are the weight matrix, the bias vector, and the input vector in the input layer, respectively. On the other hand, the decoding process of the AE can be used to calculate the reconstructed data vectors x_i':

$$S : x_i' = g(y_i w^T + b) \quad (4)$$

Where w^T, b, and y_i are the transpose weight matrix, the bias vector, and the abstract compressed data vector of the hidden layer, respectively.

The function g, used in Eq. (3) and Eq. (4), can be a linear or nonlinear activation function. In our model, a sigmoid function is used and computed as follows:

$$g(x) = 1/(1 + e^{-x}) \quad (5)$$

Once we learn the values of w^T and b by applying the AE on unlabeled data in an unsupervised learning manner, we can use them in a supervised learning manner to tune the values of w^T and b of the model, using a back-propagation algorithm with a soft-max classifier, based on the training data vectors.

B. MODEL DESCRIPTION

The proposed TSDL model is a deep learning model that consists of two stages: the initial and final decision stages. The approach used for constructing these two stages is a deep neural network (DNN), due to its performance and speed in real-time classification. In the initial stage, network traffic can be classified into normal or abnormal with a probability score value. This value is used as an additional feature to train the final decision stage for normal and multi-class attack classification.

The DNN approach, which is adopted in both stages, combines a DSAE comprising two hidden layers with a soft-max layer classifier on top. To build the proposed TSDL model, two tasks are performed in the training phase. The first is a pre-training task, where each layer of the DSAE is pre-trained individually using an unsupervised learning technique. At each layer, the error arising from reconstructing the input features is minimized.

The original unlabeled features are inputs to the first layer and the output-compressed features are inputs to the second layer. When the first and second layers are pre-trained separately, we stack them together and add a soft-max layer on top as a classifier for the next task, which is termed a fine-tuning task. In this task, a supervised learning technique based on a back-propagation algorithm is utilized to tune the parameters of the pre-trained DSAE model obtained from the previous task. Thus, our proposed model is trained by a semi-supervised learning technique. The aim of the fine-tuning task is to further minimize the prediction error by using labeled features. We briefly summarize the algorithmic steps of the training phase in two stages of our TSDL model in Algorithm 1 and Algorithm 2. The variables used in these algorithms are presented in Table 1.

After the training phase is completed, the TSDL model is able to classify unseen network traffic instances. Figure 3 shows the steps of the trained TSDL model during the classification phase. The first stage is intended to reduce the overfitting problem and mitigate bias towards normal traffic, by placing more focus on abnormal traffic, in order to classify the other types of attacks. The proposed model also provides a discriminative, abstract feature space to differentiate between normal and abnormal network traffic flows.

Algorithm 1 Initial Decision Stage Algorithm

```
Input: $x, t, lx, lt$
Output: TMDNNI, $pcl$ and $pv$.
Begin
//Initializing the parameters
  $initializeParameters(p)$;
//Reading Train and Test feature sets
  $readData\ (x, t, lx, lt)$ ;
//Pre-training task to learn the feature representation
  $y \leftarrow preTrain(x, \mathrm{IN1}, \mathrm{H1})$;
  $v \leftarrow preTrain(y, \mathrm{IN2}, \mathrm{H2})$;
//Fine-tuning task to tune the parameters
  DSAEI $\leftarrow stackedLayers\ (\mathrm{H1}, \mathrm{H2})$ ;
  DNNI $\leftarrow stackedLayers\ (\mathrm{IN}, \mathrm{DSAEI}, \mathrm{OT})$ ;
  TMDNNI $\leftarrow fineTune(x, lx, \mathrm{DNN1})$;
//Classification
  $[pcl, pv] \leftarrow classify(\mathrm{t}, \mathrm{TMDNNI})$;
  returnTMDNNI, $pcl, pv$
End
```

TABLE 1. Notations used in Algorithms 1 and 2.

Notation	Description
x, t	Training and testing sets.
lx, lt	Training and testing labels.
y, v	Abstract features of first and second hidden layers.
fc	Concatenated features.
pcl	Predicted label of normal and abnormal traffic.
$pclAllattcks$	Predicted labels of attack types.
pv	Probability value of normal and abnormal class label.
p	List of parameters.
H1,H2	First and second hidden layers.
IN1,IN2	Input layers of two Auto-encoders.
IN,OT	Input and output layers of deep neural network.
DSAEI	Deep stack auto-encoder of initial stage.
DNNI	Deep neural network of initial stage.
DSAEF	Deep stack auto-encoder of final stage.
DNNF	Deep neural network of final stage.
TMDNNI, TMDNNF	Trained models of DNNI and DNNF.

V. EXPERIMENTS AND EVALUATION

To evaluate and validate the proposed TSDL model for a NIDS, two publicly available datasets have been used. These are the benchmark KDD99 dataset and the new, complex UNSW-NB15 dataset. In the next subsections, the description of the two datasets, the preprocessing step and the comparative results are presented in detail. The preprocessing step consists of normalization and resampling of the datasets so they can be efficiently and effectively handled with the TSDL model.

A. TOOL DESCRIPTION

The experiments are conducted using MATLAB version R2015a on a laptop, with a 2.0 GHz Intel Core i7-4510U processor, 8 GB of RAM and a 64-bit Windows 10 operating system.

B. PERFORMANCE METRICS

To evaluate the performance of the TSDL model, the same evaluation measures are adopted as in most previous research on NIDSs. Specifically, literature review shows that accuracy, precision, recall, F-measure, and false alarm rate (FAR) metrics have been widely used to evaluate the effectiveness of most intrusion detection systems [45]. These metrics are

Algorithm 2 Final Decision Stage Algorithm

```
Input: $x, t, lx, lt, pv$, TMDDNI
Output: TrainedDNN1 and $pclAllattcks$
Begin
//Initializing the parameters
    $initializeParameters(p)$;
//Reading Train and Test feature sets
    $readData\ (x, t, lx, lt, pv,$ TMDDNI) ;
//Features concatenation
    $fc \leftarrow concatFeature(x, pv)$;
//Pre-training task to learn the feature representation
    $y \leftarrow preTrain(fc,$ IN1, H1);
    $v \leftarrow preTrain(y,$ IN2, H2);
//Fine-tuning task to tune the parameters
    DSAEF $\leftarrow stackedLayers$ (H1, H2) ;
    DNNF $\leftarrow stackedLayers$ (IN, DSAEF, OT) ;
    TrainedDNNF $\leftarrow fineTune(fc, lx,$ DNNF);
//Classification
    $[pcl, pv] \leftarrow classify$(t, TMDNNI);
    $fc \leftarrow concatFeature(t, pv)$;
    $[pclAllattcks] \leftarrow classify(fc,$ TMDNNF);
    returnTMDNNF,$pclAllattcks$
End
```

computed using the following equations:

$$Accuracy\ (ACC) = (TP + TN)/(TP+TN+FP+FN) \tag{6}$$

$$Precision(P) = TP/(TP + FP) \tag{7}$$

$$Recall(R) = TP/(TP + FN) \tag{8}$$

$$F - measure = 2 \times ((P \times R)/(P + R)) \tag{9}$$

$$FalseAlarmRate(FAR) = FP/(FP + TN) \tag{10}$$

where TP, TN, FP, and FN are true positives, true negatives, false positives, and false negatives, respectively.

Furthermore, the execution time of the proposed model is computed to evaluate its efficiency for real-time detection.

C. DATASETS DESCRIPTION

A number of publicly available datasets have been used for NIDS evaluation. However, they do not truly represent current real-world network traffic attacks. Assessing the degree of realism of these datasets for NIDSs is also a challenging task [26], [50]. Therefore, several evaluation metrics are used by researchers to measure the degree of realism, in an attempt to produce more representative datasets containing scalable, active behaviors of normal and abnormal traffic [50], [51]. This is imperative for the reliability and credibility of any developed NIDS. In this work, an older benchmark, namely the KDD99 dataset, and a new benchmark, the UNSW-NB15 dataset, are used to evaluate the TSDL model. The results of both benchmark datasets are analyzed in terms of accuracy and detection rate, to evaluate the effectiveness of our proposed model.

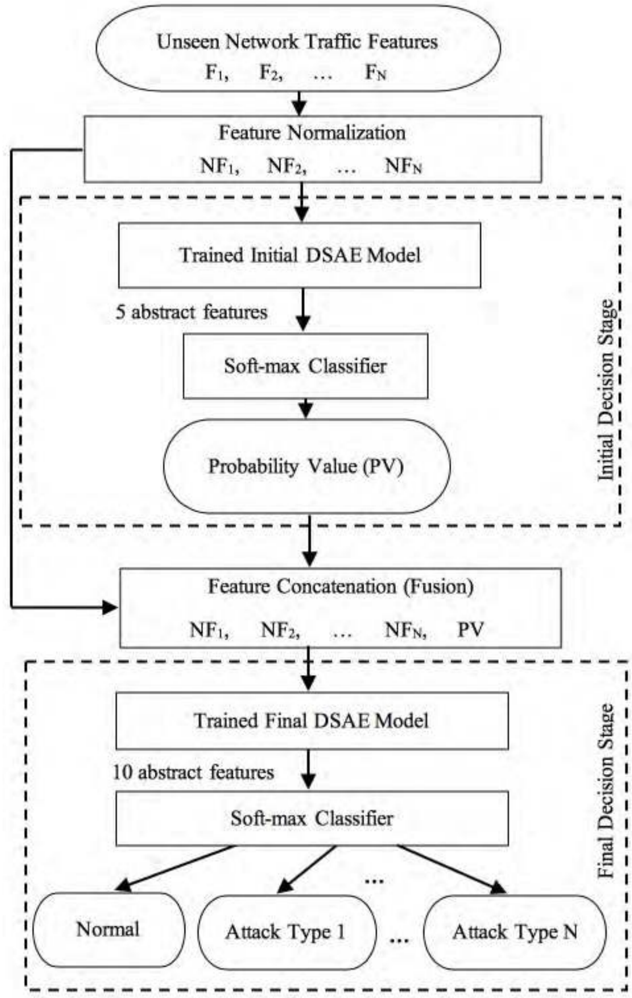


FIGURE 3. Overview of the classification phase based on the TSDL model.

The KDD99 dataset is widely used for NIDS performance evaluation [52]. The normal and attack behaviors in the KDD99 dataset are outdated and do not have any complex variations on current network traffic, which makes the evaluation of any NIDS misleading [26]. This might explain the reasons behind the high accuracy rates attained by various existing models. Additionally, this dataset includes a set of redundant records [30] and produces unbalanced distribution among the different classes of traffic [53]. It comprises about 5 million records, including abnormal and normal events collected over a number of weeks [53]. Each record consists of 41 continuous and nominal features, as well as a type label, which determines the class of traffic: normal or attack. We can find 22 different types belonging to one of four popular attacks, namely the Remote-to-Local (R2L) attack, User-to-Root (U2R) attack, Probing attack (probe) and Denial of Service (DoS) attack. Three sets of features exist in the KDD99 dataset: traffic features, content features and basic features. Traffic features are calculated by examining the connections established in the past two seconds, and are divided into two different groups, i.e., the host features (such as *rerror_rate*, *serror_rate*,etc.) and service features (such as *srv_rerror_rate*, *srv_serror_rate*, etc.) [50]. Content features allow the detection of suspicious behavior (such as *num_compromised*, *Num_failed_logins*, *logged_in*, etc.). Basic features are attributes extracted from TCP/IP connections (such as *protocol_type*, *duration*, *service*, *flag*, *dst_bytes, scr_bytes,*etc.). Besides these features, there is also an attack category label field for each instance in the dataset, named *attack_cat*. This field represents either the normal state or the attack category name. For differentiating the normal instances from abnormal instances, we add another field, *label,* which takes a value of 1 for abnormal traffic and 0 for normal traffic.

UNSW-NB15 is a newer dataset collected at the Australian Centre for Cyber Security (ACCS) by the cyber security research group [54]. A raw data of 100 GB was collected using TCP dump and Ixia PerfectStorm tools representing normal and modern network traffic attacks. The data was generated over two different simulation periods of 15 hours and 16 hours, respectively. There are approximately 2.5 million records within the UNSW-NB15 dataset. A total of 49 features were extracted using Bro-IDS, Argus tools, and some newly developed algorithms. The features of this dataset are divided into five sets: time features, content features, flow features, basic features, and additional originated features. In addition to these features, there are two labels: *attack_cat*, which is either the normal state or the attack category name, and *label*, which is 1 for abnormal traffic and 0 for normal traffic. There are nine types of attacks in the UNSW-NB15 dataset, including Worms, Shellcode, Reconnaissance, Analysis, Generic, Backdoor, DoS, Exploits, and Fuzzers [23].

Each dataset is partitioned into two distinct parts, a training dataset and a testing dataset, to assess the performance of any system developed for network intrusion detection [55]. Since the original size of the KDD99 is too large [23], [52], 10% of the KDD99 [56] is usually used to evaluate the proposed model. For the UNSW-NB15 dataset, a partition from the original dataset was selected as a training set (UNSW_NB15_training-set.csv) [57], which is also used for model evaluation. This partition has 45 features including the record ID and two class labels. Table 2 shows different classes of the 10% of KDD99 dataset used for training, and Table 3 shows the different classes of the UNSW-NB15 training dataset.

D. DATA PREPROCESSING

The data-preprocessing step includes two main tasks: the nominal-to-numeric data conversion and data-resampling tasks. First, nominal-to-numeric data conversion is applied on the two datasets. In the KDD99 dataset, there are 38 numeric features, three nonnumeric features, and one nonnumeric attack category label. Meanwhile, the UNSW-NB15 dataset contains 39 numeric features, 3 nonnumeric features, and 1 nonnumeric attack category label.

Since the proposed model only deals with numeric features, the nonnumeric features are converted into numeric values. For instance, the ’*protocol_type*’ feature contains 3 types of attributes, ’*icmp*’, ’*tcp*’, and ’*udp*’. These attributes are converted into numeric integer values, 1, 2, and 3, respectively. In the same way, the *’service’* feature, the ’*flag*’ feature and the ’*attack_cat*’ label are also represented by numeric integer values. After this, the data-resampling task is performed to solve the problems of redundant records and imbalanced class distribution. By analyzing the two datasets, we noticed that the KDD99 dataset includes a large number of redundant instances [30], [53]. However, this issue does not exist in the case of the UNSW-NB15 dataset [23]. Redundant instances may lead to the problem of imbalanced class distribution, which is known to bias a machine learning classifier towards the majority class [30], [53]. Imbalanced class distribution is a key problem resulting either from repeated instances, as in the case of KDD99 (see Figure 4), or a large difference in the number of instances of each class, as in the case of UNSW-NB15 (see Figure 5). Repeated instances in the KDD99 dataset make the class distribution of normal and abnormal records imbalanced, and the large difference in the number of normal and abnormal records in the UNSW-NB15 dataset leads to the same problem. Therefore, we down-sample the KDD99 dataset by eliminating repeated records, obtaining a sum of 145,586 unique records out of 494,021 total instances (see Table 2). For the UNSW-NB15 dataset, we used the Synthetic Minority Over-sampling Technique (SMOTE) described in [58] to up-sample the instances of normal and abnormal classes and make the distribution more balanced, as shown in Figure 5 and Table 3.

TABLE 2. Data preprocessing of KDD99 dataset.

Class Name	Original size	Original distribution (%)	New size	New distribution (%)
Normal	97,278	19.69	87,832	60.33
DoS	391,458	79.24	54,572	37.48
Probe	4,107	0.83	2,131	1.46
R2L	1,126	0.23	999	0.69
U2R	52	0.01	52	0.04
Total	494,021	100	145,586	100

TABLE 3. Data preprocessing of UNSW-NB15 dataset.

Class Name	Original Size	Distribution (%)	New size	New distribution (%)
Analysis	2,000	1.14	2,000	0.8463
Backdoor	1,746	1	1,746	0.7388
DoS	12,264	6.99	12,264	5.1895
Exploits	33,393	19.04	33,393	14.1302
Fuzzers	18,184	10.37	18,184	7.6946
Generic	40,000	22.81	40,000	16.9260
Normal	56,000	31.94	116982	49.5009
Reconnaissance	10,491	5.98	10,491	4.4393
Shellcode	1,133	0.65	1,133	0.4794
Worm.	130	0.07	130	0.0550
Total	175,341	100	236323	100

E. FEATURE NORMALIZATION

Feature normalization is a method utilized to normalize the independent features into a specific range. In the case of

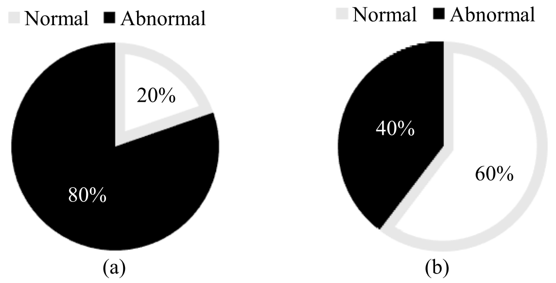


FIGURE 4. Data preprocessing of KDD99 dataset; (a) original distribution of normal and abnormal instances, and (b) new distribution of normal and abnormal instances.

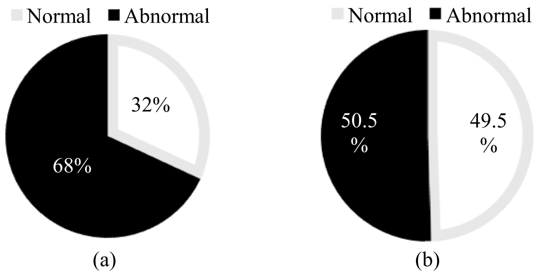


FIGURE 5. Data preprocessing of USNW-NB15 dataset; (a) original distribution of normal and abnormal instances, and (b) new distribution of normal and abnormal instances.

data processing, it is commonly known as data normalization. There are some features obtained from the network traffic that have a very large difference between the minimum and the maximum values. To suppress the effect of these outliers, a min-max scaling method is used to linearly normalize the feature values in the range of 0 and 1, according to Eq. (11).

$$f_{i,j} = \frac{f_{i,j} - \min(f_{i,j})}{\max\left(f_{i,j}\right) - \min(f_{i,j})} \tag{11}$$

where $f_{i,j}$ represents the value of the feature in row i and column j of the dataset.

F. RESULTS AND COMPARISONS

After implementing the proposed TSDL model in MATLAB programing language, we tested it on the selected datasets with a 10-fold cross validation method. Using this method, each dataset is divided into ten parts, one of which is used as a testing dataset while the remaining nine are used as a training dataset. This method is repeated 10 times, and the results are then averaged to yield a single estimation. The advantage of this evaluation is that all observations are employed for both training and testing, and each observation can be used exactly once for testing the trained model. In this context, we demonstrate the results of each stage of the TSDL model. The results of the evaluation measures are demonstrated for both initial and final decision stages. The initial stage is responsible for normal and abnormal state classification, whereas the final decision stage is used for multi-class classification (normal state and other types of attacks). This implies that the TSDL model is a flexible intrusion detection model with two options, as requested by the user.

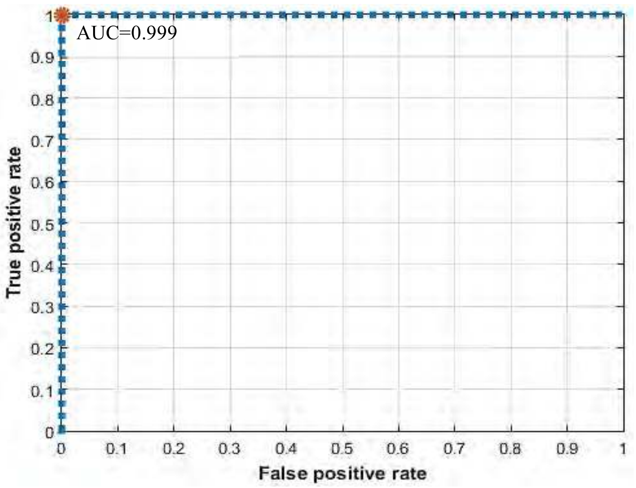


FIGURE 6. Area under curve of normal and abnormal classification of KDD99 dataset.

TABLE 4. Confusion matrix of normal and abnormal classification for KDD99 dataset.

	Normal	Abnormal	Total
Normal	**87785**	53	87838
Abnormal	47	**57701**	57748
Total	87832	57754	145586

TABLE 5. Evaluation measures of normal and abnormal classification for KDD99 dataset.

	P	R	F-measure	FAR	ACC
Normal	0.9995	0.9994	0.9993	0.0008	**99.931%**
Abnormal	0.9991	0.9992	0.9992	0.0006	
Weighted Avg.	0.9993	0.9993	0.9993	0.0007	

For the KDD99 dataset, the 41 features are normalized and reduced to 10 abstract features using the initial trained DSAE of the initial decision stage, and then classified into normal and abnormal states using a soft-max classifier. The accuracy of this stage according to the 10-fold cross validation test is 99.931%: 145486 out of 145586 instances were correctly classified. The confusion matrix and other evaluation measures are shown in Tables 4 and 5. Moreover, in order to evaluate the quality or performance of normal and abnormal detection, a Receiver Operating Characteristic (ROC) curve is shown in Figure 6. The blue dotted curve plots the true positive rates (TPRs) on the y coordinate versus the false positive rates (FPRs) on the x coordinate, and the red point on the curve represents the area under curve (AUC) value of the ROC curve. We note that the AUC has a value nearly equal to 1, which confirms that the model produces better results. In addition to Figure 6, Tables 5 and 7 demonstrate that our model attains attractive results in terms of accuracy, precision, recall and F-measure, with a very low weighted average false-alarm rate using 5 features compressed from the original ones.

TABLE 6. Confusion matrix of normal and other types of attacks classification for KDD99 dataset.

	Normal	DoS	U2R	Probe	R2L	Total
Normal	**87832**	0	0	0	0	87832
DoS	0	**54572**	0	0	0	54572
U2R	0	0	**48**	0	2	50
Probe	0	0	0	**2131**	0	2131
R2L	0	0	4	0	**997**	1001
Total	87832	54572	52	2131	999	145586

TABLE 7. Evaluation measures of normal and other types of attacks classification for KDD99 dataset.

	P	R	F-measure	FAR	ACC
Normal	1.0000000	0.9999317	1.0000000	0.0000000	**99.996%**
DoS	1.0000000	0.9998901	1.0000000	0.0000000	
U2R	0.9230769	0.9600000	0.9599873	0.0000275	
Probe	1.0000000	0.9971923	1.0000000	0.0000000	
R2L	0.9979980	0.9960040	0.9989911	0.0000138	
Weighted Avg.	0.9999588	0.9998348	0.9999788	0.0000001	

TABLE 8. Confusion matrix of normal and abnormal classification for UNSW-NB15 dataset.

	Normal	Abnormal	Total
Normal	**108540**	15874	124414
Abnormal	8442	**103467**	111909
Total	116982	119341	236323

TABLE 9. Evaluation measures of normal and abnormal classification for UNSW-NB15 dataset.

	P	R	F-measure	FAR	ACC
Normal	0.927800	0.8724	0.9262	0.0754	**89.711%**
Abnormal	0.867000	0.9246	0.8697	0.1276	

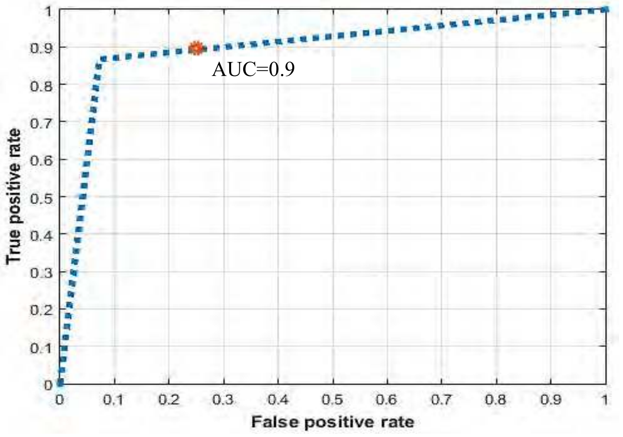


FIGURE 7. Area under curve of normal and abnormal classification of UNSW-NB15 dataset.

The normal and abnormal state classifications of the initial decision stage are based on the prediction value of the softmax classifier, which is in the range 0 to 1. This prediction value can be used as an additional feature in the final decision stage. The 41 normalized features, as well as this additional feature arising from the first stage, are reduced to 5 abstract features using the final DSAE model in the second stage. These five abstract features are used to classify the normal state and other types of attacks based on the softmax classifier in the final decision stage. The accuracy of the 10-fold cross validation test is 99.996%, representing correct classification of 145580 out of 145586 instances. In Table 6, a confusion matrix shows the number of records for each class that were classified correctly. Table 7 presents the sensitivity, specificity and FAR of each class. The weighted averages of sensitivity, specificity and FAR for each class are also calculated and presented in Table 7.

Table 7 shows that the model achieves excellent results under KDD99 by using only 10 abstract features. This is due to the ability of our proposed model to compress the original features to more discriminative abstract features and reduce weak features that affect the detection rate.

Comparative experiments are also conducted on the new UNSW-NB15 dataset to demonstrate the effectiveness of our proposed TSDL model. By using the 10-fold cross validation test mode, the accuracy of the initial decision stage is shown to be 89.711%. This corresponds to 212007 out of 236323 instances being correctly classified. The confusion matrix is presented in Table 8 and other evaluation measures are shown in Table 9.

From Table 9, we can see that the model achieves consistently good results, above 89.7%, with respect to the precision, recall and F-measure, with a small average value of FAR (0.1018). The quality of detection results is illustrated by plotting the ROC graph, which is shown in Figure 7. The red point on this curve represents the AUC value at which the model produces the best detection results. We note that the value of AUC in the graph is equal to 0.9.

For classification of the normal state and other types of attacks, the accuracy is found to be 89.134%. That is, 210643 out of 236323 instances were correctly classified. The confusion matrix and other evaluation measures are listed in Tables 10 and 11. The best F-measure values are obtained for the Normal, Generic, Exploits, Fuzzers, and Reconnaissance classes, as 0.999990, 0.986158, 0.954066, 0.916957, and 0.830501, respectively. We also note that the worst F-measure values are attained by the other classes. The reason for the poor F-measure values is that these classes are in the minority compared to others—especially the Backdoor and Worm, which have a very small number of instances. Despite the problem of imbalanced class distribution in this dataset, the proposed model is seen to achieve very good results. These results prove the effectiveness and robustness of our model. To show the advantages of using the initial decision stage in terms of the effectiveness of the TSDL model, we compute the accuracy and FAR of the final stage without using the additional feature (PV). Accuracies of 98% and 87.35% are recorded for the KDD99 and UNSW-NB15 datasets, respectively. In addition, FARs

TABLE 10. Confusion matrix of normal and other types of attacks classification for UNSW-NB15 dataset.

	Anal.	Back.	DoS	Exp.	Fuzz.	Gene.	Norm.	Reco.	Shell.	Worm.	Total
Anal.	**327**	5	31	109	28	21	0	10	0	2	533
Back.	0	**0**	0	0	0	0	0	0	0	0	0
DoS	2	9	**59**	99	14	13	0	20	2	0	218
Exp.	1648	1562	11504	**30857**	2077	726	0	2602	236	114	51326
Fuzz.	1	52	282	1311	**15572**	104	0	318	160	13	17813
Gene.	0	0	20	19	8	**39106**	0	13	0	0	39166
Norm.	0	0	0	0	0	0	**116981**	0	0	0	116981
Reco.	22	118	328	963	456	25	0	**7528**	522	0	9962
Shell.	0	0	40	35	29	5	1	0	**213**	1	324
Worm.	0	0	0	0	0	0	0	0	0	**0**	0
Total	2000	1746	12264	33393	18184	40000	116982	10491	1133	130	236323

TABLE 11. Evaluation measures of normal and other types of attacks classification for UNSW-NB15 dataset.

	P	R	F-measure	FAR	ACC
Anal.	0.16350	0.01340	0.28074	0.00789	**89.134%**
Back.	0.00000	0.00000	0.00000	0.00822	
DoS	0.00480	0.00440	0.00957	0.05478	
Exp.	0.92410	0.57140	0.95407	0.01391	
Fuzz.	0.85640	0.40300	0.91696	0.01321	
Gene.	0.97770	0.61210	0.98616	0.00519	
Norm.	1.00000	0.82000	0.99999	0.00001	
Reco.	0.71760	0.24890	0.83050	0.01438	
Shell.	0.18800	0.00850	0.31627	0.00435	
Worm.	0.00000	0.00000	0.00000	0.00062	
Weighted Avg.	0.89130	0.63270	0.9085	0.00750	

TABLE 12. Results of final decision stage with and without additional feature on KDD99 and UNSW-NB15 datasets.

Method	Dataset	ACC (%)	FAR (%)
Final Decision Stage without additional feature	KDD99	98	0.15
	UNSW-NB15	87.35	1.14
Final decision stage with additional feature	KDD99	**99.996**	**0.00001**
	UNSW-NB15	**89.134**	**0.7495**

of 0.15% and 1.14% are obtained for the KDD99 and UNSW-NB15 datasets, respectively. As shown in Table 12, using the additional feature clearly improves the effectiveness of the TSDL model in terms of accuracy and FAR.

In Tables 13 and 14, we compare the TSDL model with recent work on NIDSs conducted on the KDD99 and UNSW-NB15 datasets.

The comparison is based on the number of features used, the classifier adopted in each work, the results of FAR and the accuracy. The reported results show the superiority of our model over state-of-the-art approaches in the literature. The proposed model achieves high accuracy by learning feature representations from large amounts of unlabeled training features. Using two-stage deep feature learning, our proposed model is able to detect old and new intrusion attacks and is less influenced by the presence of an unbalanced class distribution between normal and abnormal traffic on the one hand, and between the different categories of attacks on the other.

TABLE 13. Comparison results of TSDL model with state-of-the-art methods on KDD99 dataset.

Work	No. of Features	Classifier	ACC (%)	FAR (%)
Chung and Wahid [28]	6	SSO	93.3	–
Kavitha et al. [29]	7	NL	–	3.19
Lin et al. [27]	23	DT, SA	99.96	0.021
Sindhu et al. [30]	16	Neurotree	–	1.62
Mok et al. [31]	5	RELR	98.74	1.42
GALR-DT [27]	18	DT	99.90	0.105
DBN+LR [40]	5	Logistic Regression	97.90	0.5
NDAE+RF [41]	22	Random Forest	97.85	2.15
Standard MLP	42	Soft-max	95	2.85
Proposed TSDL model	**10**	**Soft-max**	**99.996**	**0.00001**

TABLE 14. Comparison results of TSDL model with state-of-the-art methods on UNSW-NB15 dataset.

Work	No. of Features	Classifier	ACC (%)	FAR (%)
Moustafa and Slay [23]	42	DT	85.56	15.78
		LR	83.15	18.48
		NB	82.07	18.56
		ANN	81.34	21.13
		EM	78.47	23.79
GALR-DT [26]	20	DT	81.42	6.39
NAWIR et al. [35]	42	AODE	83.47	6.57
Janarthanan and Zargari [36]	5	RF	81.6175	4.40
Standard MLP	42	Soft-max	81.30	21.15
Proposed TSDL model	**10**	**Soft-max**	**89.134**	**0.7495**

We also compare TSDL with the MLP, which is a shallow architecture of the deep neural network. The accuracy and FAR for MLP and TSDL models are shown in Tables 13 and 14. It can be seen that the TSDL model attains a higher accuracy and lower FAR results

TABLE 15. Average execution time of detection for one instance in milliseconds.

Stage	Average execution time (ms)
Initial decision stage for binary-classification of normal and abnormal	0.0014407
Final decision stage for multi-class classification of normal and other types of attacks	0.001932
Two-stage deep learning model (TSDL model)	0.0033727

than MLP and the state-of-the-art approaches. It provides 99.996% accuracy and a 0.00001% FAR for the KDD99 dataset and 89.134% accuracy and a 0.7495% FAR for the UNSW-NB15 dataset. Despite the complexity of the UNSW-NB15 dataset, which contains a variety of modern intrusion patterns [23], TSDL is less affected by these variations, achieving the highest accuracy when compared to state-of-the-art models available in the literature.

Before evaluating the time efficiency of our model, we assume the model is trained periodically on up-to-date network traffic features in an offline manner. Therefore, we focus on the running time of online detection (See Table 15).

In Table 15, we summarize the average execution time in milliseconds required to detect one instance for the initial decision stage, the final decision stage and both stages.

VI. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel two-stage deep learning (TSDL) model based on a deep stacked auto-encoder (DSAE) neural network, to deal with the problem of network intrusion detection. The TSDL model comprises two stages; each stage contains two hidden layers with a softmax classifier. The deep learning model is trained in a semi-supervised manner. Specifically, each hidden layer is separately pre-trained on a large set of unlabeled network traffic features, in an unsupervised manner, and then fine-tuned using labeled network traffic features. The first stage of the model, termed the initial decision stage, is used to classify the normal and abnormal states of network traffic. The user can select the deep learning model to operate only at this stage. To detect the normal state and other types of attacks, the second decision stage is employed. In the latter, our deep learning model works in a cascade manner, where the probability value of the initial stage output is utilized as an additional feature to complement the original features. This enables the final decision stage to efficiently classify various types of attacks.

Extensive experiments have been conducted on two public datasets; specifically the benchmark KDD99 and UNSW-NB15 datasets. The latter comprises more complicated types of attacks to evaluate our proposed deep learning models. In the experiments, we performed two steps, namely data preprocessing and normalization, on the features of the datasets to make them more amenable to detection. In terms of multi-class detection accuracy, TSDL achieved impressive results, up to 99.996% for the KDD99 dataset and 89.134% for the UNSW-NB15 dataset, with a low FAR. Additionally, in terms of efficiency, the execution time consumed by the proposed model is very low, which makes it appropriate for future deployment in real-time intrusion detection tasks. Comparison with state-of-the-art approaches has demonstrated the robustness of the TSDL model, which can hence serve as a future benchmark for the deep learning and network security research communities. For future work, we plan to integrate our deep learning approach with novel reinforcement [12] and multi-task [14] learning algorithms, to further optimize our proposed network intrusion detection system.

REFERENCES

[1] M. Bijone, "A survey on secure network: Intrusion detection & prevention approaches," *Amer. J. Inf. Syst.*, vol. 4, no. 3, pp. 69–88, 2016.

[2] Z. Cai, Z. Wang, K. Zheng, and J. Cao, "A distributed TCAM coprocessor architecture for integrated longest prefix matching, policy filtering, and content filtering," *IEEE Trans. Comput.*, vol. 62, no. 3, pp. 417–427, Mar. 2013.

[3] K. Zheng, Z. Cai, X. Zhang, Z. Wang, and B. Yang, "Algorithms to speedup pattern matching for network intrusion detection systems," *Comput. Commun.*, vol. 62, pp. 47–58, May 2015.

[4] Y. Yu, J. Long, F. Liu, and Z. Cai, "Machine learning combining with visualization for intrusion detection: A survey," in *Modeling Decisions for Artificial Intelligence* (Lecture Notes in Computer Science), vol. 9880. Cham, Switzerland: Springer, 2016, pp. 239–249.

[5] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Proc. IEEE Symp. Secur. Privacy*, May 2010, pp. 305–316.

[6] Y. Yu, J. Long, and Z. Cai, "Network intrusion detection through stacking dilated convolutional autoencoders," *Secur. Commun. Netw.*, vol. 2017, Nov. 2017, Art. no. 4184196.

[7] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, Jan. 2015.

[8] L. Liu, L. Shao, X. Li, and K. Lu, "Learning spatio-temporal representations for action recognition: A genetic programming approach," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 158–170, Jan. 2016.

[9] A.-A. Liu, Y.-T. Su, W.-Z. Nie, and M. Kankanhalli, "Hierarchical clustering multi-task learning for joint human action grouping and recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 1, pp. 102–114, Jan. 2017.

[10] 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.

[11] A. Ahsan, H. Larijani, and A. Ahmadinia, "Random neural network based novel decision making framework for optimized and autonomous power control in LTE uplink system," *Phys. Commun.*, vol. 19, pp. 106–117, Jun. 2016.

[12] M. Mahmud, M. S. Kaiser, A. Hussain, and S. Vassanelli, "Applications of deep learning and reinforcement learning to biological data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2063–2079, Jun. 2018.

[13] Z. K. Malik, A. Hussain, and J. Wu, "Multilayered echo state machine: A novel architecture and algorithm," *IEEE Trans. Cybern.*, vol. 47, no. 4, pp. 946–959, Apr. 2017.

[14] F. Xiong *et al.*, "Guided policy search for sequential multitask learning," *IEEE Trans. Syst., Man, Cybern. Syst.*, no. 49, no. 1, pp. 216–226, Jan. 2019.

[15] X. Yang, K. Huang, R. Zhang, and A. Hussain, "Learning latent features with infinite nonnegative binary matrix trifactorization," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 6, pp. 450–463, Dec. 2018.

[16] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, 2006.

[17] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *J. Mach. Learn. Res.*, vol. 11, no. 12, pp. 3371–3408, Dec. 2010.

[18] J. Wu, Y. Zhang, and W. Lin, "Good practices for learning to recognize actions using FV and VLAD," *IEEE Trans. Cybern.*, vol. 46, no. 12, pp. 2978–2990, Dec. 2016.

[19] F. Kuang, W. Xu, and S. Zhang, "A novel hybrid KPCA and SVM with GA model for intrusion detection," *Appl. Soft Comput.*, vol. 18, pp. 178–184, May 2014.

[20] R. R. Reddy, Y. Ramadevi, and K. V. N. Sunitha, "Effective discriminant function for intrusion detection using SVM," in *Proc. Int. Conf. Adv. Comput., Commun. Inform. (ICACCI)*, Sep. 2016, pp. 1148–1153.

[21] W. Li, P. Yi, Y. Wu, L. Pan, and J. Li, "A new intrusion detection system based on KNN classification algorithm in wireless sensor network," *J. Elect. Comput. Eng.*, vol. 2014, Jun. 2014, Art. no. 240217.

[22] B. Ingre and A. Yadav, "Performance analysis of NSL-KDD dataset using ANN," in *Proc. Int. Conf. Signal Process. Commun. Eng. Syst.*, Jan. 2015, pp. 92–96.

[23] N. Moustafa and J. Slay, "The evaluation of network anomaly detection systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set," *Inf. Secur. J., Global Perspective*, vol. 25, nos. 1–3, pp. 18–31, 2016.

[24] N. Farnaaz and M. A. Jabbar, "Random forest modeling for network intrusion detection system," *Procedia Comput. Sci.*, vol. 89, pp. 213–217, Jan. 2016.

[25] J. Zhang, M. Zulkernine, and A. Haque, "Random-forests-based network intrusion detection systems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 5, pp. 649–659, Sep. 2008.

[26] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Comput. Secur.*, vol. 70, pp. 255–277, Sep. 2017.

[27] S.-W. Lin, K.-C. Ying, C.-Y. Lee, and Z.-J. Lee, "An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection," *Appl. Soft Comput.*, vol. 12, no. 10, pp. 3285–3290, 2012.

[28] Y. Y. Chung and N. Wahid, "A hybrid network intrusion detection system using simplified swarm optimization (SSO)," *Appl. Soft Comput.*, vol. 12, no. 9, pp. 3014–3022, 2012.

[29] B. Kavitha, S. Karthikeyan, and P. S. Maybell, "An ensemble design of intrusion detection system for handling uncertainty using Neutrosophic Logic Classifier," *Knowl.-Based Syst.*, vol. 28, pp. 88–96, Apr. 2012.

[30] S. S. S. Sindhu, S. Geetha, and A. Kannan, "Decision tree based light weight intrusion detection using a wrapper approach," *Expert Syst. Appl.*, vol. 39, no. 1, pp. 129–141, 2012.

[31] M. Mok, S. Sohn, and Y. Ju, "Random effects logistic regression model for anomaly detection," *Expert Syst. Appl.*, vol. 37, no. 10, pp. 7162–7166, 2010.

[32] J. A. Khan and N. Jain, "A survey on intrusion detection systems and classification techniques," *Int. J. Sci. Res. Sci., Eng. Technol.*, vol. 2, no. 5, pp. 202–208, 2016.

[33] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 2, pp. 1153–1176, 2nd Quart., 2016.

[34] J. Wu, W. Zeng, and F. Yan, "Hierarchical Temporal Memory method for time-series-based anomaly detection," *Neurocomputing*, vol. 273, pp. 535–546, Jan. 2018.

[35] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Multi-classification of UNSW-NB15 dataset for network anomaly detection system," *J. Theor. Appl. Inf. Technol.*, vol. 96, no. 15, 2018.

[36] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *Proc. IEEE 26th Int. Symp. Ind. Electron. (ISIE)*, Jun. 2017, pp. 1881–1886.

[37] S. M. Erfani, S. Rajasegarar, S. Karunasekera, and C. Leckie, "High-dimensional and large-scale anomaly detection using a linear one-class SVM with deep learning," *Pattern Recognit.*, vol. 58, pp. 121–134, Oct. 2016.

[38] U. Fiore, F. Palmieri, A. Castiglione, and A. De Santis, "Network anomaly detection with the restricted Boltzmann machine," *Neurocomputing*, vol. 122, pp. 13–23, Dec. 2013.

[39] 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. (BIONETICS)*, New York, NY, USA, May 2016, pp. 21–26.

[40] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep learning approach for network intrusion detection in software defined networking," in *Proc. Int. Conf. Wireless Netw. Mobile Commun. (WINCOM)*, Oct. 2016, pp. 258–263.

[41] Z. Wang, "The applications of deep learning on traffic identification," BlackHat, Tech. Ref., 2015.

[42] 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.

[43] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proc. IEEE 15th Int. Conf. Mach. Learn. Appl.*, Anaheim, CA, USA, Dec. 2016, pp. 195–200.

[44] 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.

[45] K. K. Nguyen, D. T. Hoang, D. Niyato, P. Wang, D. Nguyen, and E. Dutkiewicz. (2017). "Cyberattack detection in mobile cloud computing: A deep learning approach." [Online]. Available: https://arxiv.org/abs/1712.05914

[46] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.

[47] H. Bourlard and Y. Kamp, "Auto-association by multilayer perceptrons and singular value decomposition," *Biol. Cybern.*, vol. 59, nos. 4–5, pp. 291–294, 1988.

[48] P. Baldi and Z. Lu, "Complex-valued autoencoders," *Neural Netw.*, vol. 33, no. 8, pp. 136–147, 2012.

[49] W.-C. Lin, S.-W. Ke, and C.-F. Tsai, "CANN: An intrusion detection system based on combining cluster centers and nearest neighbors," *Knowl.-Based Syst.*, vol. 78, pp. 13–21, Apr. 2015.

[50] W. Haider, J. Hu, J. Slay, B. Turnbull, and Y. Xie, "Generating realistic intrusion detection system dataset based on fuzzy qualitative modeling," *J. Netw. Comput. Appl.*, vol. 87, pp. 185–192, Jun. 2017.

[51] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, 2012.

[52] S.-H. Kang and K. J. Kim, "A feature selection approach to find optimal feature subsets for the network intrusion detection system," *Cluster Comput.*, vol. 19, no. 1, pp. 325–333, 2016.

[53] S. Elhag, A. Fernández, A. Bawakid, S. Alshomrani, and F. Herrera, "On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on intrusion detection systems," *Expert Syst. Appl.*, vol. 42, no. 1, pp. 193–202, 2015.

[54] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proc. IEEE Mil. Commun. Inf. Syst. Conf. (MilCIS)*, Nov. 2015, pp. 1–6.

[55] A. A. Aburomman and M. B. I. Reaz, "A novel SVM-kNN-PSO ensemble method for intrusion detection system," *Appl. Soft Comput.*, vol. 38, pp. 360–372, Jan. 2016.

[56] (2018). *KDD Cup 1999 Data*. Accessed: Dec. 12, 2018. [Online]. Available: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

[57] UNSW-NB15 Dataset. (2015). *UNSW Canberra at the Australian Defense Force Academy*, Canberra, Australia. Accessed: Dec. 12, 2018. [Online]. Available: https://www.unsw.adfa.edu.au/australian-centre-for-cybersecurity/cybersecurity/ADFA-NB15-Datasets/

[58] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

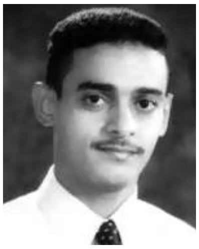
[59] A. J. Malik, W. Shahzad, and F. A. Khan, "Network intrusion detection using hybrid binary PSO and random forests algorithm," *Secur. Commun. Netw.*, vol. 8, no. 16, pp. 2646–2660, Nov. 2015.

[60] Y. Tian, M. Mirzabagheri, S. M. H. Bamakan, H. Wang, and Q. Qu, "Ramp loss one-class support vector machine; A robust and effective approach to anomaly detection problems," *Neurocomputing*, vol. 310, pp. 223–235, Oct. 2015. doi: 10.1016/j.neucom.2018.05.027.

[61] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for network intrusion detection," *Cluster Comput.*, vol. 21, no. 1, pp. 667–680, 2018. doi: 10.1007/s10586-017-0971-8.

[62] 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.

FARRUKH ASLAM KHAN received the M.S. degree in computer system engineering from the GIK Institute of Engineering Sciences and Technology, Pakistan, and the Ph.D. degree in computer engineering from Jeju National University, South Korea, in 2003 and 2007, respectively. He also received professional training from the Massachusetts Institute of Technology, New York University, IBM, and other professional institutions. He is currently an Associate Professor with the Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia. He is also Founding Director of the Wireless Networking and Security Research Group, National University of Computer and Emerging Sciences, Islamabad, Pakistan. He has over 80 publications in refereed international journals and at conferences. He has coorganized several international conferences and workshops. He has successfully supervised four Ph.D. students and 16 M.S. thesis students. Several M.S. and Ph.D. students are currently working under his supervision. His research interests include cybersecurity, body sensor networks and e-health, bio-inspired and evolutionary computation, and the Internet of Things. He is on the panel of reviewers of over 30 reputed international journals and numerous international conferences. He serves as an Associate Editor for prestigious international journals, including the IEEE Access, *PLOS One*, *Neurocomputing* (Elsevier), *Ad Hoc and Sensor Wireless Networks*, *KSII Transactions on Internet and Information Systems*, *Human-Centric Computing and Information Sciences* (Springer), and *Complex & Intelligent Systems* (Springer).



ABDU GUMAEI received the bachelor's degree in computer science from Al-Mustansiriya University, Baghdad, Iraq, and the master's degree in computer science from King Saud University, Riyadh, Saudi Arabia, where he is currently pursuing his Ph.D. degree with the Department of Computer Science. His main research interests include cybersecurity, software engineering, image processing, computer vision, and machine learning.

ABDELOUAHID DERHAB received the Engineering, master's, and Ph.D. degrees in computer science from the University of Sciences and Technology Houari Boummediene, Algiers, in 2001, 2003, and 2007, respectively. He was a full-time Researcher with the CERIST Research Center, Algeria, from 2002 to 2012. He is currently an Associate Professor with the Center of Excellence in Information Assurance, King Saud University, Riyadh, Saudi Arabia. His research interests include network security, intrusion detection systems, malware analysis, mobile security, and mobile networks.

AMIR HUSSAIN received the B.Eng. (1st Class Hons.) and Ph.D. degrees from the University of Strathclyde, Glasgow, U.K., in 1992 and 1997, respectively. He held postdoctoral and academic positions at the University of West of Scotland, from 1996 to 1998, the University of Dundee, from 1998 to 2000, and the University of Stirling, from 2000 to 2018. He joined Edinburgh Napier University, U.K., in 2018, where he is currently a Professor and Founding Head of the Cognitive Big Data and Cybersecurity Research Lab. He has (co)-authored more than 350 papers, including over a dozen books and around 150 journal papers. He has led major national, EU, and internationally funded projects and supervised over 30 Ph.D. students. Among other distinguished roles, he is General Chair of the IEEE WCCI 2020 (the world's largest technical event in computational intelligence, comprising IJCNN, the IEEE CEC, and FUZZ-IEEE) and Vice Chair of the IEEE CIS Emergent Technologies Technical Committee. He is Founding Editor-in-Chief of two leading journals, *Cognitive Computation* (Springer Nature) and *BMC Big Data Analytics*, and Springer Book Series on *Socio-Affective Computing* and *Cognitive Computation Trends*. He has also been appointed Associate Editor for various top journals, including *Information Fusion* (Elsevier), *Artificial Intelligence Review* (Springer), the IEEE Transactions on Neural Networks and Learning Systems, the *IEEE Computational Intelligence Magazine*, and the IEEE Transactions on Emerging Topics in Computational Intelligence.

• • •