Hindawi
Security and Communication Networks
Volume 2020, Article ID 7086367, 9 pages
https://doi.org/10.1155/2020/7086367

Research Article

Deep Autoencoders and Feedforward Networks Based on a New Regularization for Anomaly Detection

Marwan Ali Albahar[1] **and Muhammad Binsawad**[2]

[1]*Umm Al Qura University, College of Computers in Al-Leith, Mecca, Saudi Arabia*
[2]*King Abdulaziz University, Computer Information System Department, Jeddah, Saudi Arabia*

Correspondence should be addressed to Marwan Ali Albahar; marwanalialbahar@gmail.com

Received 23 December 2019; Revised 24 March 2020; Accepted 25 June 2020; Published 10 July 2020

Academic Editor: Leandros Maglaras

Anomaly detection is a problem with roots dating back over 30 years. The NSL-KDD dataset has become the convention for testing and comparing new or improved models in this domain. In the field of network intrusion detection, the UNSW-NB15 dataset has recently gained significant attention over the NSL-KDD because it contains more modern attacks. In the present paper, we outline two cutting-edge architectures that push the boundaries of model accuracy for these datasets, both framed in the context of anomaly detection and intrusion classification. We summarize training methodologies, hyperparameters, regularization, and other aspects of model architecture. Moreover, we also utilize the standard deviation of weight values to design a new regularization technique. Then, we embed it on both models and report the models' performance. Finally, we detail potential improvements aimed at increasing models' accuracy.

1. Introduction

The provision of an effective and robust network intrusion detection system (NIDS) remains one of the key challenges of network security. Irrespective of technological advances in the field of NIDS, many potential solutions operate by utilizing signature-based and less-capable methods instead of an anomaly detection technique. Certain factors are linked to the hesitancy in switching, including the high cost associated with the high rate of false alarms, obstacles in the attainment of valid training data, and training data longevity. However, the reliability of conventional techniques has been proven to be limited, which subsequently leads to inaccurate and inefficient detection. In this regard, this challenge is linked to the creation of a widely accepted anomaly detection technique that is capable of reducing the limitations induced by current changes occurring within modern networks. Efficient, rapid, and effective techniques are required to deal with these issues. As such, it is important to improve effectiveness and accuracy in an in-depth manner. The analysis of NIDS must be contextually aware, and it should be more detailed in order to move toward high-level observation rather than abstract representation. Changes to behavioral attributes are required in order for this to be easily comprehensible for a network's specific element, for example, protocols, versions of the operating system, individual users, and the diverse nature of data and different types of protocols available in modern advanced networks.

This introduces eminent levels of difficulty and complications, thus representing the most crucial challenge to tracing the deviation between abnormal and normal behaviors. Due to such difficulties, it remains difficult to establish an accurate standard, which increases the domain for probable exploitation or zero-day attacks.

Recent works have highlighted the application of machine learning (ML) and other existing methods such as support vector machines (SVMs), decision trees, and Naïve Bayes for the detection of network intrusions [1].

In a broad sense, ML applications have brought efficiency and accuracy in the identification of anomalies in network traffic. However, some deficiencies remain in these

methods, such as data preprocessing requiring expert knowledge (e.g., finding important and relevant features from data) and the interaction of expert personnel being required to carry out the task. As such, this not only requires human expertise but also involves an error-prone task [2]. Likewise, a large amount of training data is also required to ensure reliable results, which is a challenging task in such a diverse and vigorous environment.

Due to these limitations, deep learning (DL) algorithms have the highest priority in modern research. DL is the advanced field of ML, which can address these limitations and can resolve problems related to shallow learning. Initially, researchers demonstrated that the layer-wise learning features of DL algorithms have either better performance or performance equivalent to that of shallow learning [3]. This process helps to analyze network data deeply, and it can efficiently identify anomalies in network traffic.

One of the main aspects of building a deep learning model is regularization. Regularization is an essential component of supervised learning; the most widely used regularization techniques are L1 and L2. The application of the penalty term is the key difference between these regularization techniques. L1 penalizes the loss function by adding the absolute value of the magnitude of coefficient and thus it is suitable for feature selection or reduction, while L2 penalizes the loss function by adding the squared magnitude of the coefficient so that it gives less weights to unimportant features [3].

The main drawback of these regularizers is the dependence on the model parameters so the relationship of weight matrix entries is ignored and only a signal value of weights is controlled.

To address this drawback, we design and implement a new regularization technique as a substituted option to L1 and L2 regularizers. The new regularizer considers the dispersion of the weight values, which is known as the standard deviation, unlike L1 and L2 regularizers, in which only individual values of weights are controlled without considering the relationship among weight matrix entries.

The merit of the proposed methods lies in the adoption of new architecture for abnormal behavior detection systems.

In this paper, we present two efficient models. The first model is based on feedforward neural network (FNN) and the second model is based on a deep variational autoencoder (VAE). To reduce the error on the given training set and avoid overfitting, we introduce a new regularization technique based on taking the standard deviation of the weight matrix to get the regularization term. The motivation behind this is to create an adaptive form of weight decay. After that, we embed it in both models to study the performance of both models. We also trained our models in both semisupervised and supervised framing. Then, we conducted an in-depth analysis of the detection efficiency using different evaluation metrics. Finally, we compare our results with other well-known existing ML techniques.

Our major contributions to the existing literature are provided as follows:

(1) We present the design and implementation of two models based on VAE and FNN using a new regularization algorithm. Furthermore, we present the performance of both models on different benchmark datasets.

(2) We analyze and compare the performances of the proposed models using different evaluation measures such as accuracy, true positive rate (TPR), and F-measure with other ML methods. The experimental results show the effectiveness of the proposed models for anomaly detection.

The rest of this paper is organized as follows. In Section 2, we describe briefly the concept of feedforward neural network and the variational autoencoder. Section 3 provides the related work. In Section 4, we present the datasets used in this work. Section 5 discusses the system design and methodology. In Section 6, we give the experimental results and compare our models with other existing methods. Finally, Section 7 concludes the paper.

2. Background

2.1. Feedforward Neural Networks. FNNs are composed of various functions in a graph-like data structure that describes the connectivity among functions. The composition of functions can be denoted in the following manner.

Suppose that we have three different functions f_1, f_2, and f_3, and we let $f(x)$ be the composite of all these functions, denoted as $f(x) = f_3(f_2(f_1(x)))$. Generally, this composition describes the structure of neural networks. In neural network terminology, the aforementioned composition can be described with f_1 being the first layer, f_2 being the second layer, and so on. The number of functions in this composition is the depth of the neural network model. The final function, or the most outer function, is known as the output layer in neural network terminology.

During the training phase of the neural network model, we estimate a function $f*(x)$ to match the original unknown function $f(x)$.

The training data consists of approximate examples with target output variables $y.f*(x)$. The training examples describe the nature of the function to be estimated and specify the nature of the output layer for each data point x. In opposition to this, the training data does not describe the nature of hidden layers. This nature is decided by the learning algorithms concerning how to produce the desired output. The learning algorithms tend to estimate the behavior of hidden layers on training data to produce optimal implementation results. This is because the training data have a hidden relationship with these layers that eventually describe these layers, and the learning algorithm must locate this relationship, which explains why these are called hidden layers.

2.2. Variational Autoencoders. Variational autoencoder (VAE) [4] is a generative model that provides a probabilistic manner for describing an observation in latent space. For unsupervised learning, this is one of the most consistent

methods, and many successful cases have been reported in image processing, [4, 5] speech recognition [6], and text generation [7].

VAEs represent a very promising method, as these methods integrate variational interpretation with the employment of neural networks as function approximates in a way that searches for the approximate posterior distribution, which can be performed with stochastic gradient descent (SGD) [8]. Moreover, they vary from the state-of-the-art autoencoders (AEs), denoising autoencoders (DAEs), and sparse autoencoders (SAEs) in that they impose a distribution over the data and hyperparameters.

As a result, VAEs have the ability to create new data once the model has been trained by sampling from this distribution. This is achieved by creating a hyperparametric description of the data that can be selected to have lower feature dimensions compared to the data. Therefore, the interpretation of this description can consider a squeezed characterization of the dataset. In the domain of anomaly detection, VAE represents a pleasing fit due to its inherent probabilistic nature.

3. Related Work

Recent research on NIDS has extensively focused on the implementation of shallow learning and ML techniques such as SVM [9, 10], K-nearest neighbours (KNN) [11], random forests (RF) [9, 10, 12], artificial neural networks (ANN) [13, 14], decision trees (DT) [11, 15–18], expectation maximization (EM), linear regression (LR), Naïve Bayes (NB) [15], neutrosophic logic (NL) [19], simplified swarm optimization (SSO) [20], PCA filtering [21], random-effects logistic regression (RELR) [22], simulated annealing (SA) [16], and neurotree [23].

A model proposed by [24] depends on the averaged one-dependence estimator (AODE) technique. This model is used for the classification of multiple classes and achieved a high FPR of 6.57% and an accuracy of 83.47% on the UNSW-NB15 dataset. Using the same dataset, a random forest (RF) classifier was used by Janarthanan and Zargari in [25]. They used five feature selections to classify and detect intrusion attacks, and their method achieved an accuracy of 81.6175% and FAR of 4.4%.

An emerging branch of ML which has received significant attention is DL. Recently, several studies have extensively employed DL in the field of network intrusion detection, which subsequently brought promising prospects to this realm. In unsupervised framings, DL methods and approaches used in the field of network anomaly detection for feature learning include restricted Boltzmann machines (RBMs), deep neural networks (DNNs), deep belief networks (DBNs), and autoencoders. Erfani et al. [26] used numerous benchmark datasets to test their model, which was based on the combination of DBNs with a linear one-class SVM. Likewise, to learn compressed features that are not in the packet payloads from specific features, Fiore et al. [27] used a discriminative RBM (DRBM) approach. The binary classification of traffic into normal and abnormal behaviors was carried out based on feeding the compressed features into a softmax classifier. An anomaly detection model based on DNNs was proposed by Javaid et al. [28]. Based on the findings of their study, they reported that DL is more effective for flow-based anomaly detection in software-defined networks (SDNs). A model based on self-taught learning (STL) was proposed by Tang et al. [29] for network intrusion detection. In their experiments, they used an NSL-KDD dataset to demonstrate the superiority of DL over different approaches in terms of accuracy and performance. To recognize network traffic from raw data, Wang [30] proposed a DL approach based on a stacked autoencoder. Based on their results, it was demonstrated that that method accomplished a high performance. Furthermore, a DL approach based on recurrent neural networks (RNNs) was proposed by Yin et al. [31] for intrusion detection. The authors applied their method on the NSL-KDD dataset to measure its effectiveness. Consequently, they demonstrated the effectiveness of this DL method over traditional ML approaches for intrusion detection.

A DL method based on a DBN of RBMs having four hidden layers to reduce the feature sizes was proposed by Alrwashdeh and Purdy [32]. During the fine-tuning phase, they updated the weights of DBNs, and LRs were used to perform the classification task. They tested their proposed algorithm on the KDD Cup 1999 dataset and achieved an accuracy of 97.9% with an FPR of 0.5%. A DL-based nonsymmetric deep autoencoder (NDAE) approach was used by Shone et al. [33]. In that study, the KDD Cup 1999 and NSL-KDD datasets were used for testing purposes in combination with an RF classifier, achieving an accuracy of 97.85% and 85.42% for KDD Cup and NSL-KDD datasets, respectively. However, the FPR values were 2.15% and 14.58% for the KDD Cup and NSL-KDD datasets, respectively, which were alarming. Therefore, this method cannot be used in real-time scenarios for attack detection due to inherent deficiencies and general ineffectiveness.

A novel method based on the combination of hybrid feature selection and two-stage metaclassifier for intrusion detection was proposed in [34]. The authors used NSL-KDD and UNSW-NB15 to evaluate their model performance. Finally, they claimed that their proposed model achieved an accuracy of 91.27% and FPR of 8.90%. The authors in [35] proposed an improved anomaly-based intrusion detection system using gradient boosted machine (GBM). They used three datasets, NSL-KDD, UNSW-NB15, and GPRS, to evaluate their model using either hold-out method or tenfold cross-validation. In addition, they reported that their model yielded higher detection performance than other IDS models. An efficient DL approach based on an STL framework was proposed by Al-Qatf et al. [36]. Furthermore, a stacked autoencoder based on a two-phase DL model with a softmax classifier was proposed by Khan et al. [37]. Based on the Apache Spark framework, a DBN for feature selection and an SVM-based ensemble approach were used in [38]. This method is efficient enough to provide satisfactory detection results in large-scale networks.

4. Data

4.1. NSL-KDD Dataset. The NSL-KDD essentially shares an identical structure with the old version "KDD Cup'99 dataset" and has five categories that include *normal* and 4 *types of attacks*, as well as fields for 41 features (see Table 1). This dataset was produced by Tavallaee et al. with the intention of addressing some of the inherent problems in the KDD 99 [39]. However, the NSL-KDD dataset continues to suffer from issues and cannot be considered as a perfect representation of real networks [40]. Nevertheless, many studies in the realm of anomaly detection still use this dataset. Thus, we believe that the NSL-KDD dataset remains a valid benchmark because it contains reasonable record numbers of test and train sets which make the process of running the experiments on the complete set affordable (*even without the need to select a small portion randomly*). Thus, the evaluation results of different research works will be consistent and comparable.

4.2. UNSW-NB15 Dataset. UNSW-NB15 is a recent and complex dataset collected by the Cyber Security Research Group (CSRG) at the Australian Centre for Cyber Security (ACCS) [41].

Initially, the amount of data was large (approximately 100 GB) and it was collected through TCP dump and Ixia PerfectStorm tools, which consist of normal and various modern attack types. The data is gathered around two periods of simulation of 15 and 16 hours, respectively. The total number of instances of this dataset is approximately 2.5 million, consisting of 42 attributes excerpted using Argus, Bro-IDS, and other advanced algorithms.

There are five feature categories in this dataset: *basic features, time features, flow features, content features, and additional derived features*. Moreover, there are two types of labels apart from the features. One is *labels*: *attack_cat*, which is normal or attack type, and the other is 1 or 0, representing normal or abnormal flow, respectively. The UNSW-NB15 dataset consists of a total of nine types of cyber attacks, which include shellcode, analysis, backdoor, exploits, worms, reconnaissance, generic, DoS, and fuzzers [15].

5. Design and Methodology

5.1. Data Preprocessing. As is the case with the majority of ML problems, there was a significant amount of data preprocessing needed to successfully learn data representation for both UNSW-NB15 and NSL-KDD datasets. Both datasets are quite large for the problem and are split into both a training set and a test set. For columns that were string values, a label encoder was applied to transform the data into the unique integer representation.

5.2. Model Architecture. For the sake of exploration, we pursue both the semisupervised and supervised framings of these datasets. The two methods utilized are FNN and deep VAE.

The FNNs are applied to the supervised context and modelled as multiclass classification with each type of attack being a different class. The hidden and input layers contain a swish activation function (1) [42] with a 5-layer topology and 512/1–256/4 unit distribution. The model gradients are updated via Nesterov-based Adam optimization (NADAM) [43] with a learning rate of .0091, $\beta_1 = 0.9$, and $\beta_2 = 0.999$. A large batch size is selected for training efficiency and to smooth out gradient updates, though recent publications have shown exceptional convergence (at least when it comes to certain problem domains) with online and local training scenarios [44]. Each layer is initialized randomly, meaning the weights of the layers are initialized by sampling uniformly from a Gaussian. Furthermore, L2 weight regularization is used to further reduce inefficient learning (2) and each layer used dropout set to a value of 0.5. A softmax activation is also used, with units equal to class number on the output layer. The loss function used is a categorical cross entropy for 75 epochs.

$$f(X) = \text{sigmoid}(x) * x, \tag{1}$$

$$\lambda \sum_{i=1}^{k} w_i^2. \tag{2}$$

On the other hand, when training in the semisupervised context, we use an autoencoder. The theory behind autoencoders is fairly straightforward given previous knowledge with DL algorithms. Autoencoders can also be used on a variety of other very interesting problems. Among these are denoising image data, dimensionality reduction, and even compression [45]. For our problem, the autoencoder learns a compressed representation of the data. Since we are operating in the domain of anomaly detection, we train the model on normal data only (therein lies the difference from binary classification). The model then predicts whether or not an arbitrary input fits the learned representation. The autoencoder is trained on the entire feature set. The encoder and decoder are composed of four layers with an encoding dimension of 256 units. The number of units is halved at each subsequent layer in the encoder, with the inverse being true for the decoder. A total of 41 input and output units are used to learn data representation. Compared to the classification network, the autoencoder uses ReLU activation (3). Initialization occurs in the same manner as the first network. We optimize via NADAM (Algorithm 1) and substitute mean squared error with mean squared logarithmic error. The autoencoder uses both L1 and L2 regularization methods (see (4)).

$$f(X) = \max(0, x), \tag{3}$$

$$\lambda \sum_{i=1}^{k} |w_i| + \lambda \sum_{i=1}^{k} w_i^2. \tag{4}$$

5.3. New Regularization Technique. Regularization is a key component of supervised learning, so we embed our new regularization technique into both models as a substituted

Table 1: Attack classes based on different attack types.

Attack class	Attack types
Dos	Back, land, Neptune, pod, smurf, teardrop, mailbomb, pro-cesstable, udpstorm, Apache2, worm
Probe	Psweep, nmap, portsweep, Satan, mscan, saint
U2R ps	Buffer-overflow, loadmodule, perl, rootkit, sqlattack, xterm,
R2L	Fpt-write, guess-passwd, imap, multihop, phf, spy, warez-master, xlock, xsnoop, snmpguess, snmpgetattack, httptun- nel, sendmail, named

$$
\begin{aligned}
&g_t \longleftarrow \nabla_{\theta_{t-1}} f(\theta_{t-1}) \\
&\widehat{g} \longleftarrow (g_t / 1 - \textstyle\prod_{i=1}^{t} \mu_i) \\
&m_t \longleftarrow \mu m_{t-1} + (1-\mu) g_t \\
&\widehat{m}_t \longleftarrow (m_t / 1 - \textstyle\prod_{i=1}^{t+1} \mu_i) \\
&n_t \longleftarrow \nu n_{t-1} + (1-\nu) g_t^2 \\
&\widehat{n}_t \longleftarrow (n_t / 1 - \nu^i) \\
&\overline{m}_t \longleftarrow (1-\mu_t)\widehat{g}_t + \mu_{t+1}\widehat{m}_t \\
&\theta_t \longleftarrow \theta_{t-1} - \eta (\overline{m}_t / \sqrt{\widehat{n}_t + \xi})
\end{aligned}
$$

Algorithm 1: Nesterov-accerative adoptive moment estimation

option to L1 and L2 regularizers to observe if any improvement occurs on learning data representation. The new regularizer considers the dispersion of the weight values, which is known as the standard deviation. The new regularization technique uses the standard deviation of the weights to the loss function instead of absolute values and squared magnitude values. Thus, it restrains the learning model from taking widespread values from the weight space. The mathematical formulation of the new regularizer is given in equations (5) and (6).

$$\curlywedge \sum_{i=1}^{n} \sigma(w), \tag{5}$$

where n is the number of rows in weight matrix and i is the i-$^{\text{th}}$ row of the weight matrix. σ denotes standard deviation of weight values as given below:

$$\sigma(w) = \sqrt{\frac{1}{kn}\left\{\sum_{i=1}^{kn} w_i^2 - \frac{1}{kn}\left(\sum_{i=1}^{kn} w_i\right)^2\right\}}. \tag{6}$$

The parameter λ is used to control the values of the weight matrix, and k denotes the size of the weight vector. Particularly, it presents the number of columns in a particular weight matrix *(k depends on the features number in the dataset)*. w_i are the values of the model's weights.

5.4. Evaluation Protocols. To evaluate our models, the performance of all classifiers is evaluated in terms of accuracy, false positive rate (FPR), true positive rate (TPR), precision, and F-Score, which are calculated based on the mathematical representation given in equations (7)–(11), respectively.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}, \tag{7}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}}, \tag{8}$$

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{9}$$

$$\text{Pre} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \tag{10}$$

$$F - \text{score} = \frac{\text{Pre} + \text{TPR}}{\text{Pre} \times \text{TPR}}, \tag{11}$$

Where TP, TN, FP and FN denote true positives, true negatives, false positives and false negatives, respec-tively.

6. Results

To carry out simulations, a machine having core-i7 processor with 16 GB of RAM and 64 bit linux operating system is used. The implementation is done in python 2.7 with *KERAS* which uses an end-to-end machine learning platform called *TENSORFLOW* at backend. The training and testing time for both models on each dataset is shown in Table 2.

Due to various optimization and cutting-edge methods, both models achieved results equally, near to, or better than those of previous state-of-the-art methods for this problem. Each model is trained on both NSL-KDD and UNSW-NB15 datasets by using train-test split method and explicitly tested on the test set provided by the dataset *KDDTest+* and *UNSW_NB15_testing – set* (75% for training and 25% for validation). We have shown the results in terms of accuracy, FPR,

Table 2: Training and testing time for each model.

Model	Dataset	Training time (sec)	Testing time (sec)
Proposed FFN	NSL-KDD	700.5	436.01
Proposed VAE	NSL-KDD	1203.7	545.3
Proposed FFN	UNSW-NB15	423.8	264.2
Proposed VAE	UNSW-NB15	689.8	360.1

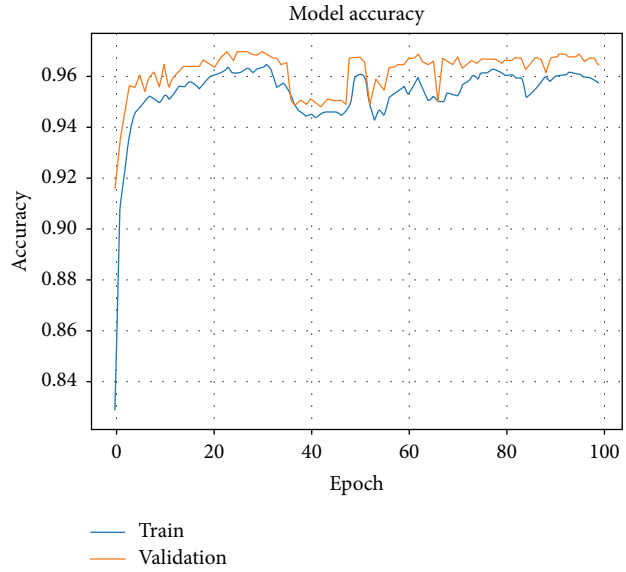

Figure 1: Training and validation accuracy on NSL-KDD dataset using FFN model.

TPR, precision, and F-Score which give shreds of evidence that the intrusion detection mechanism of the proposed methods is more than satisfactory.

High values of F-Score represent that the precision value of both models is also accurate and efficient to detect and find out anomaly from the network traffic. For our first model, the classifier converges to +95% on the validation data after approximately 50 epochs and starts to diverge after approximately 75. Even given the extremely accurate model, there remains room for further improvement.

The autoencoder converged to the same accuracy on the validation set (approximately 70 epochs) and diverged shortly thereafter. The autoencoders with embedded regularizers oscillated slightly more chaotically during training, likely due to both the difference in problem and feature set when compared to the classifier (along with different hyperparameters, regularizers, and activation functions).

Likewise, after embedding the new regularizer to both aforementioned models, we observed up to 1.7% improvement in average validation accuracy.

The feedforward model was trained for 100 epochs, and the average testing accuracy achieved through feedforward model is 96.7% and 94.7% for NSL-KDD and UNSW-NB15 datasets, respectively. The progressions for training and validation accuracy for FFN models on both datasets are shown in Figures 1 and 2, respectively. The accuracy with the new regularizer is significantly better than other regularizers as shown in Tables 3 and 4.

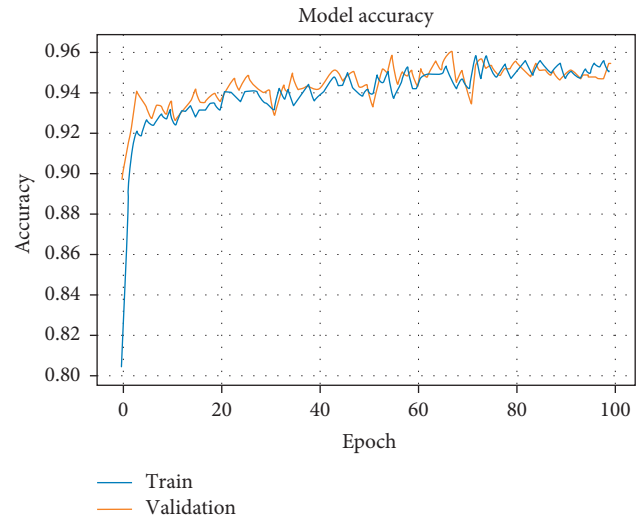

Figure 2: Training and validation accuracy on UNSW-NB15 dataset using FFN model.

Similarly, for VAE models with embedded regularizer, we achieved average testing accuracy of 97.01% and 93.3% for NSL-KDD and UNSW-NB15 datasets, respectively. The progressions for training and validation accuracy for VAE models on both datasets are shown in Figure 3 and Figure 4, respectively. The corresponding average accuracies with other performance measures such as FPR, TPR, precision

Table 3: Comparison of the proposed models' results with existing approaches used on the UNSW-NB15 dataset.

Model	No. of features	Classifier	Accuracy	FPR	TPR	Precision	*F*-score
Proposed FFN	42	DNN	94.7	1.04	94.24	86.76	89.75
Proposed VAE	42	Deep VAE	93.3	0.93	95.21	87.9	90.2
Two-stage ensemble [34]	—	Two-stage meta	91.72	8.90	91.30	91.60	—
GALR-DT [11]	20	DT	81.42	6.39	—	—	—
NAWIR [24]	42	AODE	83.47	6.57	98.5	—	—
[25]	5	RF	81.61	4.40	81.6	—	79.5
Standard MLP [37]	42	Softmax	81.30	21.15	—	—	—
		Softmax	89.13	0.74	63.27	—	—
		NB	82.07	18.56			
TSDL [37]	10	DT	85.56	15.78			
		ANN	81.34	21.13			
		LR	83.15	18.48			
[15]	42	EM	78.47	23.79	—	—	—

Table 4: Comparison of the proposed models' results with existing methods on the NSL-KDD dataset.

Model	Accuracy	FPR	TPR	Precision	F-score
Proposed FFN	96.7	0.64	95.86	88.2	90.96
Proposed VAE	97.01	0.83	95.42	87.9	91.3
Two-stage ensemble [34]	85.79	11.7	86.8	88.0	—
DBN [33]*	80.58	19.42	80.58	—	84.08
S-NDAE	85.42	14.58	85.42	—	87.37
SVM [38]*			86.22		89.30
Ensemble			90.45		93.91
Multilayer			91.98		94.36
DBN + SVM	—	—	92.17	—	94.65
STL-IDS [36]	80.48	—	76.56	—	79.07

*All the models in the same cell belong to the same reference.

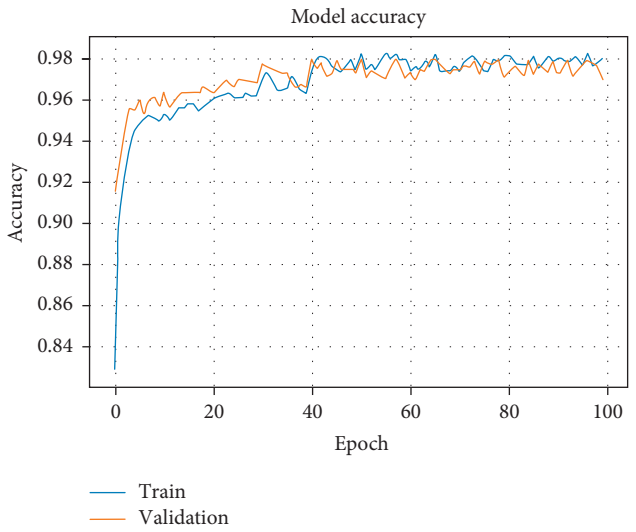

Figure 3: Training and validation accuracy on NSL-KDD dataset using VAE model.

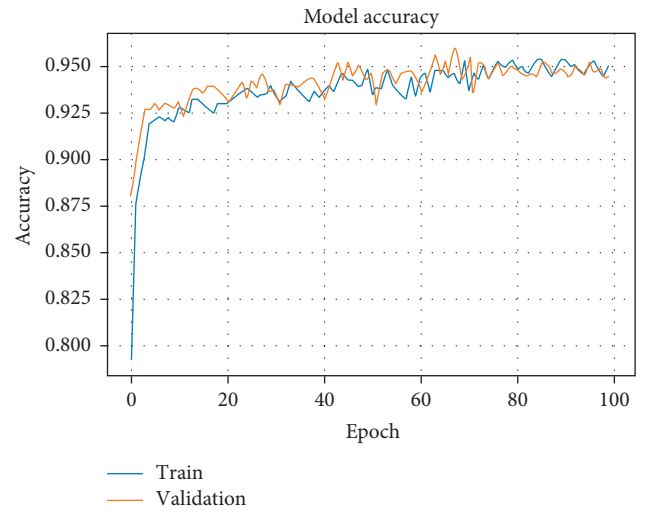

Figure 4: Training and validation accuracy on UNSW-NB15 dataset using VAE model.

and F-Score are also computed and shown in Table 3 and Table 4.

From Figures 1–4, it is evident that the accuracy converges to some extent after 80th epoch but still it oscillates between certain values. Upon investigating, it is found that the test data provided is well shuffled and balanced. So, the instances of each class were distributed approximately equally. This gives us two advantages. The first one is that the model does not overfit (in which the validation accuracy is less than the training accuracy). The second advantage is that the data is well generalized.

If we notice FPR and TPR from Tables 3 and 4, the FPR is high, while the TPR is good. The reason behind that is that if we notice equations (8) and (9), the FPR incorporates the true negatives (TNs). So, the FPR is inversely proportional to TNs. So, when TN is high, the FPR value will be low and vice versa. It means that the model counts in some samples of the other classes in the class it is training for. Similarly, the model falsely rejects some samples of the class it is training for. In other words, in equation (9), the TPR is inversely proportional to FNs.

7. Conclusion

We introduce the design and the implementation of two models employing a new regularization technique that meets or exceeds previous bests on the NSL-KDD and UNSW-NP15 datasets for both classification and anomaly detection domains. The new models are tested on several datasets available in network security domain (i.e., NSL-KDD and UNSW-NB15). The simulation results represent that the performance of new models is better than those of other methods. However, there are many different ways in which one could alter model optimization to further increase test accuracy. Firstly, more could be done with data prepossessing and feature selection. For example, one could implement PCA to extract principle components, use a low variance filter, or find a different method to select important features. Additionally, with domain knowledge, one could engineer more features that may increase model effectiveness. Regarding the models, a key area could include hyperparameter tuning either manually or via an algorithm. Furthermore, experimenting with other regularizers, optimizers, and activation functions could also be worth investigation. Overall, our proposed models performed well and have resulted in satisfactory performance measures compared to existing state-of-the-art methods. For comparison purposes, Tables 3 and 4 present the performances of prior existing methods and the proposed models experimented in this paper.

Data Availability

Datasets used to support the findings of this study are included within the article.

Ethical Approval

This article does not contain any studies with human participants performed by any of the authors.

Conflicts of Interest

The authors declare that there are no conflicts of interest.

Acknowledgments

This work was supported by the Deanship of Scientific Research (DSR), King Abdulaziz University, Jeddah, under Grant no. (D-432-611-1441). The authors, therefore, gratefully acknowledge DSR's technical and financial support.

References

[1] B. Dong and X. Wang, "Comparison deep learning method to traditional methods using for network intrusion detection," in *Proceedings of the 2016 8th IEEE International Conference on Communication Software and Networks (ICCSN)*, IEEE, Beijing, China, pp. 581–585, June 2016.

[2] R. Zhao, R. Yan, Z. Chen, K. Mao, P. Wang, and R. X. Gao, "Deep learning and its applications to machine health monitoring: a survey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 14, no. 8, pp. 1–14, 2016.

[3] P. Murugan and S. Durairaj, "Regularization and optimization strategies in deep convolutional neural network," http://arxiv.org/abs/1712.04711.

[4] DP. Kingma and M. Welling, "Auto-encoding variational bayes," 2013, http://arxiv.org/abs/1312.6114.

[5] DJ. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," 2014, http://arxiv.org/abs/1401.4082.

[6] W.-N. Hsu, Y. Zhangand, and J. Glass, "Unsupervised domain adaptation for robust speech recognition via vari- ational autoencoder-based data augmentation," 2017, http://arxiv.org/abs/1707.06265.

[7] S. Semeniuta, A. Severyn, and E. Barth, "A hybrid convolutional variational autoencoder for text generation," 2017, http://arxiv.org/abs/1702.02390.

[8] C. Doersch, "Tutorial on variational autoencoders," 2016, http://arxiv.org/abs/1606.05908.

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

[10] 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, 2008.

[11] C. Khammassi and S. Krichen, "A GA-LR wrapper approach for feature selection in network intrusion detection," *Computers & Security*, vol. 70, pp. 255–277, 2017.

[12] A. J. Malik, W. Shahzad, and F. A. Khan, "Network intrusion detection using hybrid binary PSO and random forests algorithm," *Security and Communication Networks*, vol. 8, no. 16, pp. 2646–2660, 2015.

[13] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: learn- ing useful representations in a deep network with a local denoising criterion," *Journal of Machine Learning Research*, vol. 11, no. 12, pp. 3371–3408, 2010.

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

[15] 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," *Global Perspective*, vol. 25, no. 1-3, pp. 18–31, 2016.

[16] 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," *Applied Soft Computing*, vol. 12, no. 10, pp. 3285–3290, 2012.

[17] 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, 2018.

[18] A. J. Malik and F. A. Khan, "A hybrid technique using binary particle swarm optimization and decision tree pruning for

network intrusion detection," *Cluster Computing*, vol. 21, no. 1, pp. 667–680, 2018.

[19] B. Kavitha, D. S. Karthikeyan, and P. Sheeba Maybell, "An ensemble design of intrusion detection system for handling uncertainty using neutrosophic logic classifier," *Knowledge-Based Systems*, vol. 28, pp. 88–96, 2012.

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

[21] 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, 2015.

[22] M. S. Mok, S. Y. Sohn, and Y. H. Ju, "Random effects logistic regression model for anomaly detection," *Expert Systems with Applications*, vol. 37, no. 10, pp. 7162–7166, 2010.

[23] 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.

[24] M. Nawir, A. Amir, N. Yaakob, and O. B. Lynn, "Multi-classification of UNSW-NB15 dataset for network anomaly detection system," *Journal of Theoretical and Applied Information Technology*, vol. 96, no. 15, 2018.

[25] T. Janarthanan and S. Zargari, "Feature selection in UNSW-NB15 and KDDCUP'99 datasets," in *Proceedings of the IEEE 26th Int. Symp. Ind. Electron*, pp. 1881–1886, June 2017.

[26] 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 Recognition*, vol. 58, pp. 121–134, 2016.

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

[28] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference BioInspired Information Communication Technology*, BIONETICS), New York, NY, USA, pp. 21–26, May 2016.

[29] 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 *Proceedings of the International Conference Wireless Network Mobile Communication (WINCOM)*, pp. 258–263, October 2016.

[30] Z. Wang, "The applications of deep learning on traffic identification," BlackHat, Technical Report, 2015.

[31] 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.

[32] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *Proceedings of the IEEE 15th International Conference Mach. Learn. Appl.*, pp. 195–200, Anaheim, CA, USA, December 2016.

[33] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 2, no. 1, pp. 41–50, 2018.

[34] B. A. Tama, M. Comuzzi, and K.-H. Rhee, "TSE-IDS: a two-stage classifier ensemble for intelligent anomaly-based intrusion detection system," *IEEE Access*, vol. 7, pp. 94497–94507, 2019.

[35] B. A. Tama and K.-H. Rhee, "An in-depth experimental study of anomaly detection using gradient boosted machine," *Neural Computing and Applications*, vol. 31, no. 4, pp. 955–965, 2019.

[36] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi, "Deep learning approach combining sparse autoencoder with SVM for network intrusion detection," *IEEE Access*, vol. 6, pp. 52843–52856, 2018.

[37] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, *TSDL: A Two Stage Deep Learning Model for Efficient Network Intrusion Detection*, p. 1, IEEE Access, 2019.

[38] N. Marir, H. Wang, G. Feng, B. Li, and M. Jia, "Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using Spark," *IEEE Access*, vol. 6, pp. 59657–59671, 2018.

[39] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defence Applications*, IEEE, Ottawa, Canada, pp. 53–58, July 2009.

[40] J. McHugh, "Testing Intrusion detection systems," *ACM Transactions on Information and System Security (TISSEC)*, vol. 3, no. 4, pp. 262–294, 2000.

[41] N. Moustafa and J. Slay, "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *Proceedings of the Military Communications and Information Systems Conference*, pp. 1–6, November 2015.

[42] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for activation functions," 2017, http://arxiv.org/abs/1710.05941.

[43] D. P. Kingma and J. B. Adam, "A method for stochastic optimization," 2014, http://arxiv.org/abs/1412.6980.

[44] T. Lin, S. U. Stich, and M. Jaggi, "Don't use large mini-batches, use local sgd," 2018, http://arxiv.org/abs/1808.07217.

[45] L. Theis, W. Shi, A. Cunningham, and F. Huszar, "Lossy image compression with compressive autoencoders," in *Proceedings of the International Conference on Learning Representations (ICLR2017)*, Toulon, France, April 2017.