Identification of Network Attacks Using a Deep Learning Approach

Najwa Altwaijry
ntwaijry@ksu.edu.sa
Computer Science Department
College of Computer and Information Sciences
King Saud University

Summary
Traditional network defense technologies such as firewalls are unable to detect the evolving types of attacks on networks, leading to the need for network intrusion detection systems (NIDSs) that provide better solutions. In this paper, we propose an effective deep learning method to NIDS based on a two-stage approach with a sparse autoencoder and a number of different classifiers, to create three models. The proposed approach uses the autoencoder for feature learning and dimensionality reduction, thereby reducing training and testing times. The new feature vector is then input into three classifiers, improving their detection capability for intrusion and classification accuracy. We study and compare our models with a number of other works in the literature as well as some state-of-the-art methods. Results show that our approach performs better than all other approaches in terms of detection rate, and comparably in terms of accuracy.

Keywords:
Sparse Autoencoder, Deep Learning, Anomaly Detection, NIDS.

1. Introduction

Cyber security is becoming a top priority for many countries around the world. Individuals and organizations are subject to various types of attacks, such as: malware, phishing, and denial of service attacks. According to [1], 69% of cyber security breaches in 2019 were conducted by outsiders. *Network Intrusion Detection Systems* (NIDS) are tools that monitor traffic on networks in order to detect malicious activity. NIDS analyzes all traffic to distinguish between intrusions and normal network behavior. NIDS are either Signature-based (SNIDS) or Anomaly detection-based (ADNIDS) [2]. SNIDS monitor network traffic and compare it with a pre-defined list of rules. Although very efficient in detecting *known* attacks, SNIDS are not well suited for identifying previously *unseen* attack signatures. ADNIDSs can be thought of as white lists [3]. ADNIDS detect malicious traffic that potentially deviates from normal traffic patterns. For example, in the Internet Control Message Protocol, traffic is considered abnormal when it is greater than 3% of network traffic, when normally it is only 1% [3]. Thus, ADNIDS are able to effectively detect new attack forms.

The task of network intrusion detection can be formulated as a binary classification task, where it is required to classify network traffic into either normal or abnormal traffic. The problem can also be modeled as a multi-classification problem, in which specific types of attacks are classified. ADNIDS have attracted interest from researchers in various computer science areas. Many approaches have been used in ADNIDS, such as statistical and machine learning methods [4].

Statistical methods for anomaly detection assume that data is generated in a particular distribution, however, this is not always true. Traditional machine learning methods cannot scale up to handle large amounts of data. Research suggests that commercial NIDS are not effective for todays' emerging types of attacks [5], as the increasing volume of network data requires techniques that can analyze it rapidly and efficiently.

Deep learning is a subset of machine learning algorithms that has shown success in many real-world applications. Research shows that the accuracy of deep learning architectures outperforms traditional approaches by a large margin [6]. In cyber security, deep learning has been used to tackle many problems, including: detection, modeling, monitoring, analysis, and defense against various types of network attacks [7][8]. Deep learning methods are able to capture representative features from the dataset and can generate more effective models than shallow-based approaches. According to Kim et al. [9], deep learning-based anomaly detection achieves high accuracy with the use of different deep learning architectures.
In this paper, we propose three novel deep learning model for ADNIDS, to be employed within modern computer networks. Our models are capable of analyzing a wide range of network traffic and detecting novel network attacks. Our models are composed of two components, a Sparse Autoencoder (SAE), followed by a classifier. In particular,

Manuscript received April 5, 2020
Manuscript revised April 20, 2020

this work focuses on the performance of various classifiers incorporated within our models. We evaluate our models on the well-known NSL-KDD dataset.

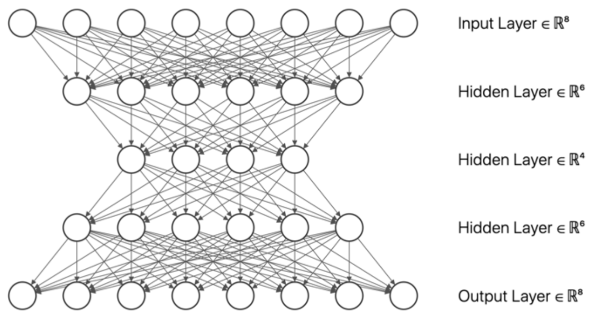
The remainder of this paper is organized as follows. In Section 2, we present the relevant background to our work. In Section 3, we discuss previous related work in NIDS. In Section 4, the proposed architecture and the dataset it is evaluated on are presented. In Section 5, we explain the experimental setup, and discuss our main findings. Finally, we conclude the study in Section 6 and provide suggestions for future work.

2. Background

Autoencoders are an unsupervised machine learning technique used for representation learning. Autoencoders compress the original input into a reduced representation, and from the reduced representation, reconstruct the original input, i.e. $h_{W,b}(x) \approx x$. This allows the autoencoder to discover correlations within the data, as in Principal Component Analysis (PCA) [10].

Sparse autoencoders are a type of autoencoder that enforce a sparsity constraint on the hidden units, thereby forcing the network to learn a representation that relies on a small number of units. The sparsity constraint may be enforced using regularization, such as L1 regularization, on the activations of hidden units, or via another method such as KL-Divergence [10].

Autoencoders generally transform the input into a lower dimensional representation (encoding), then transform the lower dimensional representation back into the input (decoding). The final encoded vector, capturing the most relevant information, may be used as an alternate lower-dimensional feature representation of the original raw data. A deep autoencoder has multiple hidden encoding layers, followed by multiple hidden decoding layers. The layers may be symmetrical or asymmetrical. The first hidden layer learns first order features, the second hidden layer learns second order hidden features from the first layer, and so on. A simple deep autoencoder is shown in Figure 1.


Input Layer $\in \mathbb{R}^8$
Hidden Layer $\in \mathbb{R}^6$
Hidden Layer $\in \mathbb{R}^4$
Hidden Layer $\in \mathbb{R}^6$
Output Layer $\in \mathbb{R}^8$
Figure 1: Deep Autoencoder

Our models are composed of two parts: a sparse autoencoder, followed by a classifier. For our classifiers, we implemented three different types:

1. *Deep Neural Network (DNN)*: a fully-connected deep neural network.
2. *Random Forest*: an ensemble learning method that groups so called “weak learners” to form a “strong learner” [11]. The weak learners in this case are individual decision trees that are then combined to form a strong learner, which is the forest.
3. *Support Vector Machine (SVM)*: a non-probabilistic linear classifier that constructs a hyperplane to classify data [12].

In Section 4, we present more details pertaining to our selected algorithms.

3. Literature Review

NIDS have been addressed using various traditional machine learning techniques in the literature, such as decision trees, SVM K-Nearest Neighbor, and Random Forests [13]. Recently, deep learning demonstrated its effectiveness for many cyber security tasks [7][8], including intrusion detection. Vinayakumar et al. [14] presented an evaluation of various deep learning algorithms for the classification of intrusions in the KDD’99 dataset. Variations and combinations of deep learning architectures, such as: Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Long short-term memory (LSTM), and Gated Recurrent Unit (GRU) were tested. The best performance for the multiclass classification problem was achieved by the combination of CNN and LSTM architectures. The simple CNN performed better than other combinations for the binary case.

Shone et al. [15] combined deep and shallow learning in order to analyze network traffic. In their proposed architecture, the authors used a Non-Symmetric Deep Auto-Encoder (NDAE) followed by a Random Forest algorithm. For unsupervised feature learning, the proposed model utilized two NDAEs arranged in a stack. Each NDAE consisted of three hidden layers. The performance evaluation of the architecture for the five-class network traffic classification tasks shows that the obtained accuracy is 97.85% for the KDD'99 dataset.

Al-Hawawreh et al. [16] proposed a deep learning-based technique for intrusion detection in industrial internet-of things systems. The proposed technique operates on information collected from TCP/IP packets. The proposed work combines deep auto-encoder and deep feedforward neural network architectures. Experimental results using NSL-KDD and UNSW-NB15 datasets showed that the proposed model can achieve a higher detection rate and lower false positive rate than similar approaches in the literature. For the five-class classification task in NSL-KDD dataset, the detection rate is 99% and the false positive rate is 1.8%.

Al-Qatf et al. [17] also took a similar approach. The authors proposed to combine a sparse autoencoder and support vector machine (SVM). The sparse autoencoder was used to reconstruct and learn the input training dataset. An SVM was used to build the NIDS model using the new training dataset. Experimental results using the NSL-KDD dataset showed that the proposed architecture accelerates the training and testing times of the SVM. For the binary classification task, the model archived 84.96% accuracy, and for the five-class classification, the model reached an accuracy of 80.48%. In both cases, the obtained results surpass those of traditional methods such as: decision trees, naïve Bayes, and SVM.

Peng et al. [18] proposed a *deep confidence neural network* and a back-propagation neural network for intrusion detection. The dataset features were extracted using a *Restricted Boltzmann machine*. The method was validated using subsets of the KDD'99 dataset. Results showed 95.45% accuracy. The study concluded that the 5-layer RBM network structure feature extraction algorithm is more suitable for feature learning tasks in high-dimensional space.

Javaid et al. [19] propose a deep learning based approach combining a sparse auto-encoder with Softmax regression. They evaluated their work on the NSL-KDD dataset, achieving an average F-score of 75.76% on the multiclass classification task. Potluri et al. used a DNN to identify abnormalities in network data, and tested their approach on the NSL-KDD dataset [20], achieving low accuracy on some classes.

The literature review shows that there is still room for improvement in detection accuracy as well as detection rate. The field of ADNIDS is young, with researchers conducting various experiments combining different algorithms for feature extraction, training and optimization, followed by classification. On the basis of this analysis, we propose a combination of SAE and three classifiers, and aim to study the performance of each classifier based on the SAE encoded features. First, we use SAE for reduced and effective feature representation of the raw NSL-KDD dataset, followed by the use of three classifiers for classification.

4. Proposed Methodology

In this paper, we propose a two-stage approach for the binary classification of network attacks. The first stage is an SAE that encodes an effective representation of the raw NSL-KDD dataset, and the second stage is a classifier that classifies input based on these features. We implement three models, each with a different classifier: a deep neural network, a random forest, and a support vector machine. We evaluate our models on the well-known NSL-KDD dataset. All classifiers were trained on KDDTrain$^+$ and tested on KDDTest$^+$.

4.1 NSL-KDD Dataset

The older KDD'99 dataset contains a large number of redundant records in both training and test datasets [21], biasing classification results. Records in the training set also appear in the testing set, inflating accuracy scores. For this reason, we use the NSL-KDD benchmark dataset. It is regarded as an improved and reduced version of the original KDD'99 dataset [21], and is widely used to assess the performance of intrusion detection systems. NSL-KDD does not contain redundant records [21][22]. There are 125,973 records in KDDTrain$^+$, and 22,544 records in KDDTest$^+$. The dataset contains the following types of network attacks: (1) Denial of Service (DoS), (2) Probing Attack, (3) User to Root Attack (U2R), and (4) Remote to Local Attack (R2L). Records in the dataset come in five classes, as follows: normal, DoS, Probe, R2L, and U2R, where the last four labels correspond to the four types of attacks. Each record in NSL-KDD is described using 41 features.

4.2 Data Preprocessing

The NSL-KDD dataset contains a number of non-numeric features, which need to be processed prior to using. We used ordinal encoding to convert non-numeric features to numeric features. Categorical features were converted into numerical values, e.g.: 'protocoltype' has three types of attributes: 'tcp', 'udp', and 'icmp' and were converted into numerical values: 1,2,3, respectively. This is repeated for all categorical features (Protocol Type, Service and Flag), thus mapping the 41-dimensional feature map with textual categorical features into a 41-dimensional feature map with numerical categorical features. Next, we scale all samples in KDDTrain$^+$ and KDDTest$^+$, as the NSL-KDD dataset's features have an extremely large gap between minimum and maximum values. Samples are scaled into a distribution with $\mu = 0$ and $\sigma^2 = 1$.

4.3 Sparse Autoencoder

As outlined in Section 2, an autoencoder is able to learn a reduced non-trivial feature vector that scales well to accommodate high-dimensional inputs. SAEs are considered effective learning algorithms for reconstructing a new feature representation in an unsupervised manner. Our models all share the same autoencoder, outlined next.

Our autoencoder is composed of two stacked autoencoders, a symmetrical and asymmetrical sparse autoencoder, respectively. The number of layers in the first autoencoder is as follows: the input layer has 41 features, followed by three encoding layers, with sizes 64, 32, and 26, respectively. This is followed by two decoding layers of sizes 32 and 64. Next, we stack the second autoencoder with encoding layers of size 32 and 26, before the final decoding layer, of size 41. All hidden layers employ the Leaky ReLU (see Equation 1) function, with $\alpha = 0.2$.

$$Leaky\ ReLU(x) = \begin{cases} \alpha x & x < 0 \\ x & x \geq 0 \end{cases} \tag{1}$$

For the sparsity constraint in our autoencoder, we employ L1 regularization on all hidden layers, with $\lambda = 1 \times 10^{-6}$. Our optimizer is Adam [23], with a learning rate of 0.001, optimizing the mean squared error function (see Equation 2), where Y_i is the ground truth, and $\hat{Y}_i$ is the predicted value for all examples i.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{2}$$

We train our autoencoder for a maximum of 150 epochs, although we employ early stopping if the loss does not improve for 10 epochs. The model is trained on 80% of the training set and validated on 20% of the training set. Finally, we extract our reduced set of features from the second autoencoder and scale these reduced features such that they fit into a distribution with $\mu = 0$ and $\sigma^2 = 1$. The final feature vector output is then used as input to our chosen classifiers. The use of a reduced feature vector reduces training and testing time for our classifiers.

4.4 Deep Neural Network

Our first classifier is a deep fully connected neural network. We call this model SAE-DNN. Our network consists of 4 hidden fully connected layers. The numbers of neurons in each hidden layer are 64, 32, 16 and 8, respectively. The hidden layers employ the ReLU function (see Equation 3).

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

A loss function is then employed to estimate the loss, in order to compare and measure the accuracy of the prediction result compared with the correct result. We calculate the loss using cross-entropy or log loss as shown in Equation 4, where y_i is the label (1 for normal traffic and 0 for anomalous traffic) and $p(y_i)$ is the predicted probability of the point being normal traffic for all n examples.

$$H_p(q) = -\frac{1}{n}\sum_{i=1}^{n} y_i \lg\big(p(y_i)\big) + (1 - y_i)\lg\big(1 - p(y_i)\big) \tag{4}$$

In order to regularize our neural network and prevent overfitting, we employ dropout, which randomly removes neurons and their connections. Dropout is employed on all layers with a probability $p = 0.3$. Each layer also regularizes its output using L2 regularization with $\lambda = 0.005$.

The output layer is used to output the final class of the current example, where we used the Sigmoid activation function, as in Equation 5, which quashes the input into the range [0,1].

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5}$$

Our deep neural network's optimizer is Adam [23], with a learning rate of 0.0008. We train our network for a maximum of 150 epochs, although we employ early stopping if the loss does not improve for 10 epochs. The model is trained on 80% of the new training set and validated on 20% of the new training set, and batch size is set to 128. All parameters were set experimentally.

4.5 Random Forest

A Random Forest is an ensemble learning method, the principle of which is to group 'weak learners' to form a 'strong learner'. Random forests have many advantages such as low bias, outlier robustness and overfitting correction [15], all of which are useful for classifying ADNIDS scenarios. We call this model SAE-RF. We train our random forest classifier using the new features extracted from the encoded representation learned by the stacked autoencoders and classify network traffic into normal and attack scenarios. We used 50 estimators with a maximum depth of 5.

4.6 Support Vector Machine

The SVM classifier produces a hyperplane to separate a class of positive instances from a class of negative instances, maximizing the margin between support vectors [24]. SVMs may be used with many functions such as Linear, Polynomial, Sigmoid, and RBF kernels. We call this model SAE-SVC. We use the RBF kernel, also known as the Gaussian kernel, in our work. In our model, we set the parameters as follows: $C = 1$ and $\gamma = \frac{1}{n \cdot Var(X)}$.

5. Experimental Results

5.1 Experimental Settings

Our models were implemented using Tensorflow [25], an open source machine learning library, utilizing Keras [26]. Experiments were carried out using GPUs running on the Google Colab [27] environment in order to minimize training time.

5.2 Performance Measures

For the evaluation of the proposed models, we consider the most important performance indicators for intrusion detection systems. The following performance measures are calculated: accuracy, precision, detection rate, and F-measure, where *TP* (true positive) indicates the number of anomaly records that are identified as anomaly, and *FP* (False positive) is the number of normal records that are identified as anomaly. *TN* (true Negative) is the number of normal records that are identified as normal, and *FN* (false negative) is the number of anomaly records that are identified as normal.

Accuracy: the percentage of records classified correctly, calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

Precision (*P*): the percentage of records correctly classified as anomaly out of the total number of records classified as anomaly. Precision is calculated as follows:

$$P = \frac{TP}{TP + FP} \tag{7}$$

Detection Rate (*DR*): also known as *Recall*, the percentage of records correctly classified as anomaly out of the total number of anomaly records. The detection rate can be calculated as follows:

$$DR = \frac{TP}{TP + FN} \tag{8}$$

F-measure (*F*): a measure that combines both precision and detection rate and is calculated as follows:

$$F = \frac{2 * P * DR}{P + DR} \tag{9}$$

5.3 Performance Evaluation

We designed our experiments to study the efficiency and effectiveness of our three chosen classifiers using SAE as a feature extractor. We evaluated our models on KDDTest$^+$ from the NSL-KDD dataset to classify traffic into either *Normal* or *Anomaly*. We compare our results with similar approaches in the literature as well as various state-of-the-art (SOTA) classification algorithms on the same dataset. In particular, we compare our model with Naïve Bayes, J48, Random Forest, Bagging, and Adaboost, which we implemented in the Waikato Environment for Knowledge Analysis (WEKA) [28]. Our models and all SOTA models were trained and validated on KDDTrain$^+$ and tested on KDDTest$^+$. KDDTrain$^+$ was split into 80% for training and 20% for validation. Table 1 presents the results of our three models compared with other models in the literature on a number of measures: Accuracy (Acc.), Precision (P), Detection Rate (DR), and F-measure (F).

Model	*Acc. %*	*P %*	*DR %*	*F %*
ANN [29]	81.20	N/A	N/A	N/A
SDN-DNN [30]	75.75	83.00	75.00	74.00
SAE-SVM [17]	**84.96**	96.23	76.57	**85.28**
RNN [31]	83.28	N/A	N/A	N/A
Naïve Bayes [28]	76.12	92.38	63.27	75.10
J48 [28]	81.53	**97.14**	69.61	81.10
Random Forest [28]	80.45	97.05	67.72	79.77
Bagging [28]	82.63	91.87	76.23	83.32
Adaboost [28]	78.44	95.28	65.37	77.54
SAE-DNN	82.00	86.00	**83.00**	82.00
SAE-RF	77.00	83.00	79.00	77.00
SAE-SVC	79.00	85.00	81.00	79.00

Table 1: Results comparison on NSL-KDD KDDTest$^+$ for Binary Classification of Network Traffic

First, we look at the performance of our three proposed classifiers. Figure 2**Figure *2*:** Performance comparison of our three classifiers shows the accuracy (Acc.), precision (P), detection rate (DR), and F-measure (F) for our classifiers. We notice that SAE-DNN performs better than the other two classifiers on all metrics. The deep neural network was able to outperform the random forest as well as the SVM.

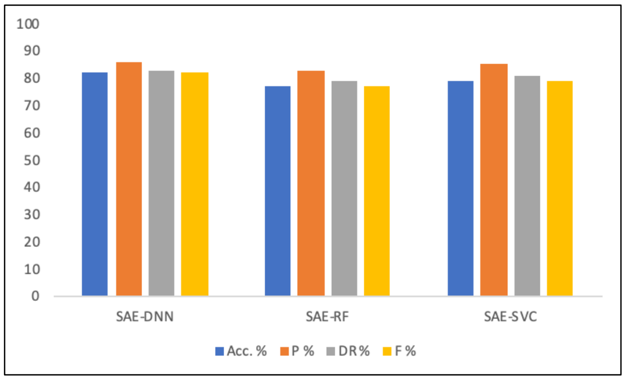


Figure 2: Performance comparison of our three classifiers

Next, we compare the three proposed classifiers to the state-of-art (SOTA) algorithms. Figure 3 shows that our three classifiers outperform all the SOTA algorithms in terms of accuracy and detection rate. The SAE-DNN model is comparable to bagging in terms of accuracy and F-measure.

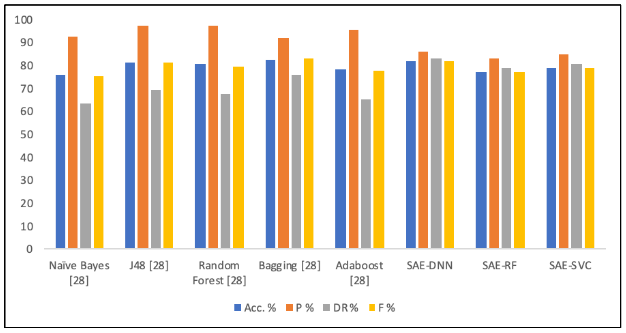


Figure 3: Performance comparison of our models compared with SOTA WEKA models

Now, we look at how our three proposed classifiers compare with similar approaches in the literature. As Table 1 shows, SAE-DNN has the best detection rate, and is able to detect 83% of all attacks, outperforming all other methods in the literature. This shows that the DNN is able to better distinguish network attacks than all other models. SAE-DNN performs better than ANN [29] and SDN-DNN [30] in terms of accuracy and F-measure. The table also shows that our proposed model SAE-DNN is very close to or better than other approaches in terms of accuracy rate, with the best detection rate.

6. Conclusion

In this paper, we have proposed three models to address the problem of ADNIDs. Our models are structured into two components: an SAE that extracts a relevant feature vector, followed by three classifiers. The SAE was used for feature learning and dimensionality reduction, and the reduced feature vector was used as input to three classifiers. The first classifier, SAE-DNN, was the best performing one, outperforming all other models in terms of detection rate, and very close in terms of accuracy. SAE-DNN outperformed all SOTA models on all measures except precision. The use of a reduced feature vector also reduced training and testing time for our classifiers. For future work, we plan to expand our model to the multiclass classification problem. We also plan to investigate different architectures for our SAE, such as a deeper structure with a smaller encoding layer size.

Acknowledgments

The authors extend their appreciation to the Deanship of Scientific Research at King Saud University for funding this work through the Research Project No R5-16-03-29.

References

[1] "2019 Data Breach Investigations Report | Verizon Enterprise Solutions." https://enterprise.verizon.com/resources/reports/dbir/.
[2] E. Aminanto and K. Kim, "Deep learning in intrusion detection system: An overview," in *2016 International Research Conference on Engineering and Technology (2016 IRCET)*, 2016.
[3] T. J. Shimeall and J. M. Spring, *Introduction to Information Security*. Elsevier, 2014.
[4] "Machine Learning Technique - an overview | ScienceDirect Topics." https://www.sciencedirect.com/topics/computer-science/machine-learning-technique.
[5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.
[6] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A review on deep learning techniques applied to semantic segmentation," *ArXiv Prepr. ArXiv170406857*, 2017.
[7] M. Alazab and M. Tang, *Deep Learning Applications for Cyber Security*. Springer, 2019.
[8] Z. Chen, K. He, J. Li, and Y. Geng, "Seq2Img: A sequence-to-image based approach towards IP traffic classification using convolutional neural networks," in *2017 IEEE International Conference on Big Data (Big Data)*, 2017, pp. 1271–1276.
[9] K. Kim and M. E. Aminanto, "Deep learning in intrusion detection perspective: Overview and further challenges," in *2017 International Workshop on Big Data and Information Security (IWBIS)*, 2017, pp. 5–10.
[10] A. Ng, "Sparse Autoencoder." CS294A Lecture notes, Stanford University.

[11] L. Breiman, “Random forests,” *Mach. Learn.*, vol. 45, no. 1, pp. 5–32, 2001.
[12] C. Cortes and V. Vapnik, “Support-vector networks,” *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, 1995.
[13] H. Liu and B. Lang, “Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey,” *Appl. Sci.*, vol. 9, no. 20, p. 4396, 2019.
[14] R. Vinayakumar, K. P. Soman, and P. Poornachandran, “Applying convolutional neural network for network intrusion detection,” in *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017, pp. 1222–1228.
[15] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, “A deep learning approach to network intrusion detection,” *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 2, no. 1, pp. 41–50, 2018.
[16] M. AL-Hawawreh, N. Moustafa, and E. Sitnikova, “Identification of malicious activities in industrial internet of things based on deep learning models,” *J. Inf. Secur. Appl.*, vol. 41, pp. 1–11, 2018.
[17] 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, doi: 10.1109/ACCESS.2018.2869577.
[18] W. Peng, X. Kong, G. Peng, X. Li, and Z. Wang, “Network Intrusion Detection Based on Deep Learning,” in *2019 International Conference on Communications, Information System and Computer Engineering (CISCE)*, Jul. 2019, pp. 431–435, doi: 10.1109/CISCE.2019.00102.
[19] 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 on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, pp. 21–26.
[20] S. Potluri and C. Diedrich, “Accelerated deep neural networks for enhanced intrusion detection system,” in *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2016, pp. 1–8.
[21] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *2009 IEEE symposium on computational intelligence for security and defense applications*, 2009, pp. 1–6.
[22] Y. Hamid, V. R. Balasaraswathi, L. Journaux, and M. Sugumaran, “Benchmark Datasets for Network Intrusion Detection: A Review.,” *IJ Netw. Secur.*, vol. 20, no. 4, pp. 645–654, 2018.
[23] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *ArXiv Prepr. ArXiv14126980*, 2014.
[24] S. R. Gunn, “Support vector machines for classification and regression,” *ISIS Tech. Rep.*, vol. 14, no. 1, pp. 5–16, 1998.
[25] “TensorFlow,” *TensorFlow*. https://www.tensorflow.org/.
[26] “Keras Documentation.” https://keras.io/.
[27] “Google Colaboratory.” https://colab.research.google.com/notebooks/intro.ipynb (accessed Mar. 29, 2020).
[28] E. Frank, M. A. Hall, and I. H. Witten, *The WEKA workbench*. Morgan Kaufmann, 2016.
[29] B. Ingre and A. Yadav, “Performance analysis of NSL-KDD dataset using ANN,” in *2015 international conference on signal processing and communication engineering systems*, 2015, pp. 92–96.
[30] 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 *2016 International Conference on Wireless Networks and Mobile Communications (WINCOM)*, 2016, pp. 258–263.
[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.

Najwa Altwaijry is an Assistant Professor of Computer Science at King Saud University. She received her PhD degree in 2014 from the College of Computer Sciences at King Saud University. Her research interests include machine learning, swarm intelligence, evolutionary computation, cyber security and bioinformatics.