Intrusion detection and classification with autoencoded deep neural network

Shahadate Rezvy[1][0000−0002−2684−7117], Miltos Petridis[1][0000−0003−1275−1023], Aboubaker Lasebae[1][0000−0003−2312−9694], and Tahmina Zebin[2][0000−0003−0437−0570]

[1] Middlesex University , London, United Kingdom
{s.rezvy, M.Petridis, A.Lasebae}@mdx.ac.uk
[2] University of Manchester, United Kingdom
{tahmina.zebin}@manchester.ac.uk

Abstract. A Network Intrusion Detection System is a critical component of every internet connected system due to likely attacks from both external and internal sources. A NIDS is used to detect network born attacks such as denial of service attacks, malware, and intruders that are operating within the system. Neural networks have become an increasingly popular solution for network intrusion detection. Their capability of learning complex patterns and behaviors make them a suitable solution for differentiating between normal traffic and network attacks. In this paper, we have applied a deep autoencoded dense neural network algorithm for detecting intrusion or attacks in network connection and evaluated the algorithm with the benchmark NSL-KDD dataset. Our results showed an excellent performance with an overall detection accuracy of 99.3% for Probe, Remote to Local, Denial of Service and User to Root type of attacks. We also presented a comparison with recent approaches used in literature which showed a substantial improvement in terms of accuracy and speed of detection with the proposed algorithm.

Keywords: Deep learning · secure computing · intrusion detection system · autoencoder · dense neural network.

1 Introduction

Over the past few decades, the Internet has penetrated all aspects of our lives. Experts predict that by 2020 there would be 50 billion connected devices. As technology becomes more and more integrated, the challenge to keep the systems safe and away from vulnerability or attacks increases. Over the years we have seen an increase in hacks in banking systems, healthcare systems and many Internet of Things (IoT) devices [1]. These attacks cause billions of dollars in losses every year and loss of systems at crucial times. This has led to higher importance in cyber security specifically in the intrusion detection systems. A related challenge with most modern-day infrastructure is that data requirements pertaining to security are often an afterthought. It is assumed that this impacts

the results of any machine learning algorithm applied towards the problem; however, an analysis contrasting the differences are yet to be seen. In addition to this, there is little research in the results of applying next level analysis using deep learning algorithms to determine if there is an improvement in accuracy versus its traditional machine learning counterparts. Deep learning (hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms. Recently, deep learning based methods have been successfully implemented in NIDS applications. Deep learning can substitute for manually designed feature extraction to create more secure firewall or intrusion detector [2].

In this work, we have proposed a data mining based hybrid intrusion detection system for distinguishing normal and intrusive events from the NSL-KDD dataset [3]. We focused on exploring low latency models while maintaining high accuracy by proposing a deep auto-encoded dense neural network (DNN) framework for effective intrusion detection. The NIDS using deep learning did alleviate the need of feature selection or feature engineering during the detection process. In our design, the autoencoder facilitated an unsupervised pre-tarining on the data to provide compressed and less noisy representation of the input space, while the final dense neural network functioned as the supervised classifier for our experimental intrusion detection scenario.

The remainder of this paper is organized as follows. Section 2 introduces the background literature in the recent development in intrusion detection systems using deep learning techniques, we then provide details on our parameter settings for the implemented model in section III. The performance of the developed model for attack classification is evaluated in section IV. We also compared the results with some recent deep learning techniques appeared in the literature using the same dataset. Finally, the paper is concluded in section V along with ideas for future work.

2 Background Literature

In recent years, machine learning has been widely applied to problems in detecting network attacks, particularly novel attacks. Given the landscape of the Internet, machine learning can be applied to handle the massive amounts of traffic to determine what is malicious or benign. NIDS are classifiers that differentiate unauthorized or anomalous traffic from authorized or normal traffic. Fig. 1 shows an illustration of the propoed components for the NIDS implementation. As can be seen in the diagram, the network gateways and routers devices could potentially host an NIDS. In the recent past, researchers have investigated a wide variety of Machine Learning techniques to design intrusion detection systems for normal and attack classification. However the NIDS design community suffered from some major challenges as there are very few labeled traffic dataset from real networks for developing an NIDS. Additionally, effective feature selection from the Network Traffic dataset for anomaly detection is difficult. The features selected for one class of attack may not work well for other categories of attacks due to continuously changing and evolving attack scenarios.

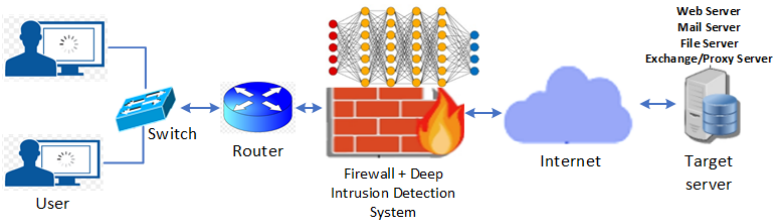

Fig. 1. Illustration of the proposed network intrusion detection system(NIDS)

Because of the lack of public data sets for network-based IDSs, we used the NSL-KDD dataset (which may not be a perfect representation of existing real networks) to compare different intrusion detection methods. However, traditional machine learning methods depend heavily on feature engineering, and extracting features is often time-consuming and complex. Thus, it is impractical to detect attacks with traditional machine learning methods in real-time applications [4]. To set our research in context, in the next subsection, we present a literature review on the intrusion detection systems that used deep learning techniques and the NSL-KDD dataset for their performance benchmarking.

Deep machine learning for intrusion detection: Neural networks have become an increasingly popular solution for network intrusion detection systems (NIDS). Their capability of learning complex patterns and behaviors make them a suitable solution for differentiating between normal traffic and network attacks [6]. One of the earliest work found in literature that used deep learning approach with Deep Belief Network (DBN) as a feature selector and Support Vector Machine (SVM) as a classifier was reported in [7]. This approach resulted in an accuracy of 92.84% when applied on training data of NSL-KDD dataset. We observed a use of Artificial neural networks (ANN) with enhanced resilient backpropagation for the design in ref [8] . This work also used the training dataset only by subdividing it in three parts (training (70%), validation (15%) and testing (15%)). The inclusion of the unlabeled data for testing resulted in a reduction of performance for this method. Ref [9] applied fuzzy clustering based ANN that resulted above 80% detection accuracy of with a low false positive rate. In [10], the work used an unsupervised greedy learning algorithm to learn similarity representations over the nonlinear and high-dimensional data in KDD dataset.The results show that four-hidden-layer Restricted Boltzmann machines can produce the higher accuracy in comparison with SVM and ANN. A deep Neural Network is essentially a multilayer perceptron, which was initially developed by stacking linear classifiers. The model is fed inputs, inputs get multiplied by weights and the passed into an activation function. The model uses backpropagation to adjust weights and increase accuracy. Deep Neural Networks with three or more hidden layers support higher generalization capability in comparison to ANN [11]. In [12], a deep belief network for malicious code detection method was reported that included an autoencoder based dimensionality reduction technique. An accelerated deep architecture was introduced in Potluri et al. [13] to iden-

tify the abnormalities in the network data. They evaluated the performance of the model training related to different processor types and number of cores. Li et al. [14] presented an intrusion detection method using convolutional neural networks which adopts a novel representation learning method of graphic conversion. However, the model accuracy ranged from 79% to 81.57% which did not justify the increased latency added by the representation learning of the feature space. In reference [15], Vinaykumar et al. reported various convolutional and recurrent neural network architectures for IDS implementation, however their results for under-represented remote to local (R2L) and user to root (U2R) attacks were very low. Shone et al.[1] reported a stacked non-symmetric deep autoencoders with random forest classifier achieved an overall accuracy of 85.42%, with very poor performance on the R2L and U2R intrusion categories. A stacked autoencoder based implementation of NIDS by Farahnakian et al. [16] produced high accuracy (94.71%) and high detection rate (94.53%), however, these models could still be improved. The findings from our literature review have shown that despite the high detection accuracy being achieved, with most researchers still experimenting on combining various algorithms (e.g. training, optimisation, activation and classification) and layering approaches to produce the most accurate and efficient solution for a specific dataset.

In our research, we focused on exploring low latency models while maintaining high accuracy by proposing a hybrid deep neural network that includes an unsupervised pre-training using autoencoders to make the model more adaptive to the changes in the network traffic. We then used a dedicated supervised dense neural network structure for the final classification. In our design, we made sure the memory or processing power to train and execute machine learning models are within the capability of the routers processing power. We hence believe the model and work presented in this paper will serve towards the real time and low latency implementation of the NIDS models.

3 Data Pre-processing and Implementation of the model

In this section, we further discuss on the technical details of our proposed deep learning based approach to increase the performance accuracy of the NSL-KDD benchmarking dataset. In the dataset, there are 41 different nominal, numeric and binary features extracted from a specific network connection. The 42^{nd} column contains data about the label on the network access type(i.e intrusion or normal). Table 1 summarizes the types of attacks or intrusion events available in the dataset. We have remapped the entire dataset to a five class scenario with four attack classes and one normal category for training and classification purposes. Further description on the features extracted in the dataset can be found in ref [17], [18].

3.1 Data Pre-processing

The KDD dataset [5] has a huge number of redundant records , which causes the learning algorithms to be biased towards the frequent records, and thus

prevent them from learning infrequent records which are usually more harmful to networks such as U2R and R2L attacks [18].

In this work, we have merged the training and test dataset from the KDD 1999 dataset and removed all duplicate records in data. We then intentionally added some duplicates in the R2L and U2R attack groups to have a increased representation of the smaller attack groups in the dataset. This in total constituted in 151063 instances of labelled data. We then separated 20% of the entire dataset as test set, the remaining 80% of the data was used for training and validation purpose.

We performed the following pre-processing on the NSL-KDD training and test sets:

- Encoding: As mentioned previously, the dataset contains different features with different value ranges. Hence, The dataset is preprocessed before applying autoencoded learning on it. Before using the data-set, we first convert the categorical features to numeric values. Categorical features such as protocol type, service and flag are encoded into numbered variables using one-hot (1-to-n) encoding. We have also applied log encoding on the large numerical features such as source bytes, destination bytes and duration to avoid any kind of biasing on the training due to their large values.
- Scaling and Normalization: All numeric features values are ranged between 0 and 1. We performed a min-max normalization on the feature vectors.

The output labels are one hot encoded. Therefore, the total number of input dimension is 122 after performing the above-mentioned steps and output dimension is 5 (4 attacksor intrusion types and 1 normal).

3.2 Proposed Model

Fig. 2 illustrates the work flow and the proposed deep model architecture for the intrusion detection system. Similar to most existing deep learning research, our proposed classification model was implemented using python keras library [19] with TensorFlow back end. All of our evaluations were performed on a Windows machine with an Intel Xeon 3.60GHz processor, 32 GB RAM and an NVIDIA GTX 1050 GPU.

We employed two main functional stages in our proposed model. An autoencoder based unsupervised pre-training layer and a supervised dense neural network for classification of the attack types for the NIDS. We describe our intuition for using these components in the system development in the coming subsections.

Unsupervised pre-training with Autoencoder An autoencoder is a type of artificial neural network used to learn efficient data representation in an unsupervised manner. In our proposed model, we have employed an autoencoder with an encoding and a decoding layer that has been trained to minimize the reconstruction error. This incorporated prior knowledge from the training set to

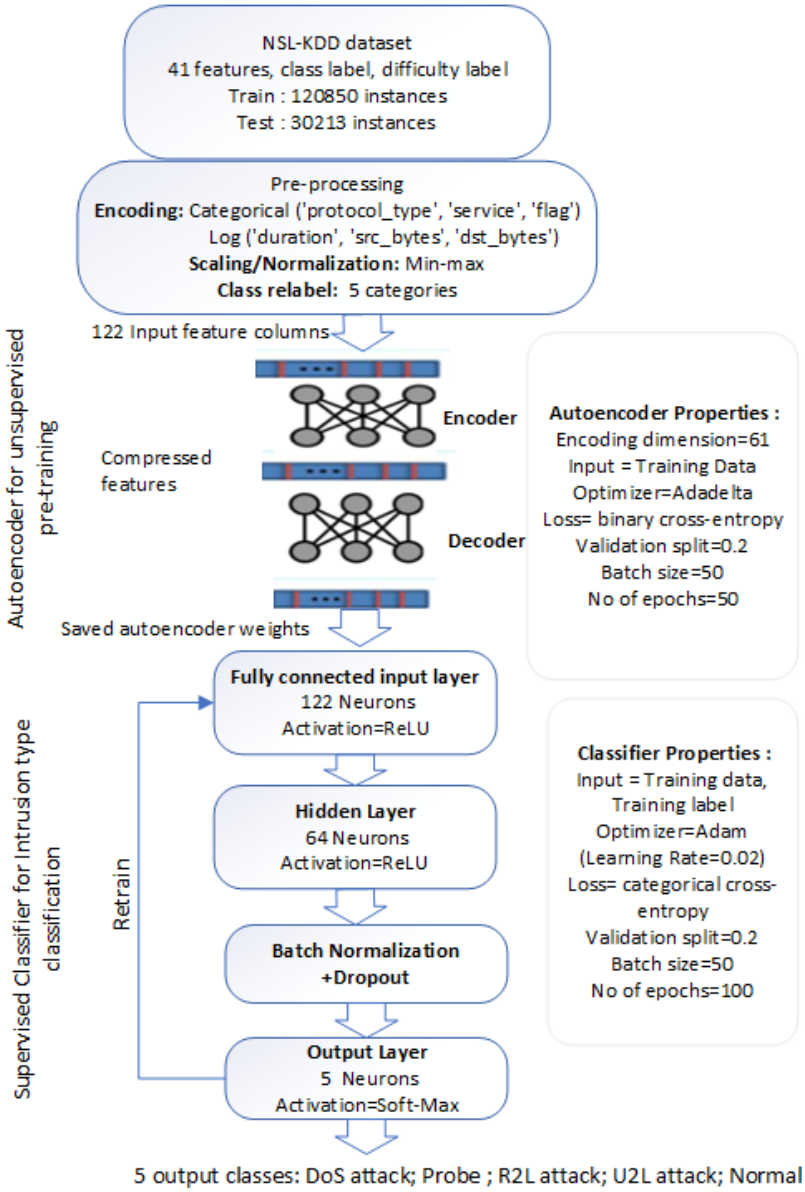

Fig. 2. Workflow and architecture of the proposed autoencoded dense neural network

effectively learn from the data itself and provide good performance. Such pre-training allow both the data for the current task and for previous related tasks to self-organize the learning system to build the learning system in a data driven fashion. We have fed the autoencoder with the features from the training dataset without labels (unsupervised). A set of compressed and robust feature is built at the end of this step. The encoder part of the autoencoder aims to compress input data into a low-dimensional representation, and there is a decoder part that reconstructs input data based on the low-dimension representation generated by the encoder.

For a given training dataset $X = \{x_1, x_2, ..., x_m\}$ with m samples or instances, where x_n is an n-dimensional feature vector, the encoder maps the input vector x_n to a hidden representation vector h_n through a deterministic mapping f_θ as given in (1)

$$h_n = f_\theta(x_n) = \sigma(Wx_n + b) \tag{1}$$

where W is a $p \times p$, p is the number of hidden units, b is a bias vector, θ is the mapping parameter set $\theta = \{W, b\}$. σ is sigmoid activation function.

The decoder maps back the resulting hidden representation h_n to a reconstructed p-dimensional vector y_n in input space.

$$y_i = g_\theta(h_n) = \sigma(Wh_n + b) \tag{2}$$

The goal of training the autoencoder is to minimize the difference between input and output. Therefore, a error function is calculated by the following equation:

$$E(x, y) = \frac{1}{m} \left\| \sum_{i=1}^{m} (x_n - y_n) \right\|^2 \tag{3}$$

The main objective is to find the optimal parameters to minimize the difference between input and reconstructed output over the whole training set (m).

Supervised Classification with DNN A three layer dense neural network is employed of a is trained by using the first auto-encoder,s output as inputs. This task sequence is retrained in a supervised manner with the class labels and the input feature given to the classifier. We have used a softmax activation layer as the output layer. The layer calculates the loss between the predicted values and the true values, and the weights in the network are adjusted according to the loss.

The simple *softmax* layer, which is placed at the final layer, can be defined as follows:

$$P\left(c|x\right) = \text{argmax}_{c \in C} \frac{\exp(x_{L-1} W_L + b_L)}{\sum_{k=1}^{N_C} \exp(x_{L-1} W_k)}, \tag{4}$$

where c is the number of classes, L is the last layer index, and N_C is the total number of class types including normal network connection and intrusion. After this stage, all layers are fine-tuned through back-propagation in a supervised

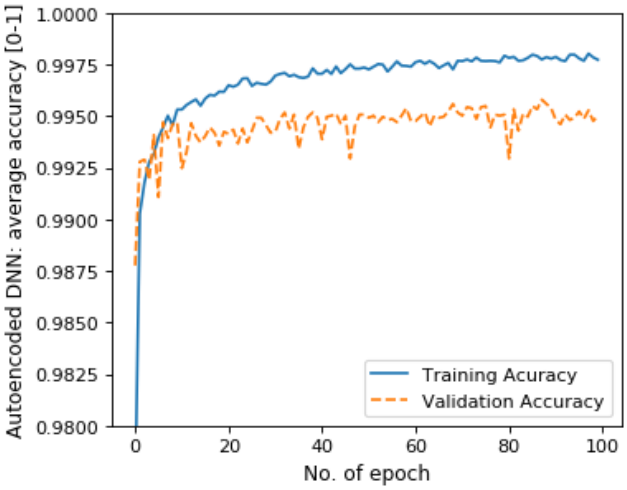

Fig. 3. Training and validation set accuracy over increasing number of epochs(training iterations)

way. In the test phase, the softmax layer outputs the probability of the predicted categories.

Fig. 3 plots the performance of the network training process for 100 iterations. During training, we have used additional techniques such as dropout and batch normalization to avoid over fitting and also to speedup the training process. The proposed algorithm achieves approximately 99% accuracy for the training set in 20 iterations which is four times faster if no dropout and batch normalization was employed. We used a five-fold cross-validation using 20% of the training data and the validation data set also achieves high overall accuracy with slight deviation over time. Potentially this allows a reduction in the training epochs required, and will be of vital importance for developing low-latency models and training future networks with bigger data sets. The proposed algorithm is summarized here in Algorithm I.

4 Model Evaluation

The main aim of our designed model is to show that the intrusion detection system we designed using deep autoencoded neural network framework will maximize accuracy and minimize any falsely categorized values.

Algorithm 1: Auto-encoded DNN training algorithm

Input: Taining dataset $X = \{x_1, x_2, ..., x_m\}$, Number of layers L

for $l \in [1, L]$ do;

Initialize $W_l = 0, W_l = 0, b_l = 0, b'_l = 0$;

Encoding layer;

Calculate encoding or hidden representation using equation(1);

$h_l = s(W_l x_{l1} + b_l)$;

Decoding layer;

while not loss==stopping criteria do;

Compute y_l using equation (2);

Compute the loss function: binary cross-entropy;

Update layer parameters $\theta = \{W, b\}$;

end while;

end for;

Classifier:Dense neural network, Soft-max activation at the output layer;

Initialize (W_{l+1}, b_{l+1}) at the supervised layer;

Calculate the labels for each sample x_n of the training dataset X;

Apply batch normalization and dropout for speeding up the calculation;

Perform back-propagation in a supervised manner to tune parameters of all layers, loss function categorical cross-entropy;

end;

Output: Class labels

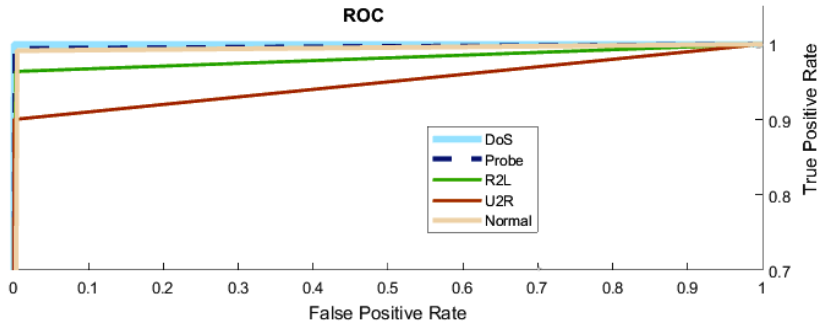

Fig. 4. Receiver operating characteristics for different classes

Table 1. Attack types in the KDD Cup 1999 dataset [5]

Attack Type	Description of the Attack	Categories
Denial of Service (DoS)	A DoS attack is a type of attack in which the hacker makes a computing resources or memory too busy or too full to serve legitimate networking requests and hence denying users access to a machine.	apache, back, land, mailbomb, neptune, pod, processtable, smurf, teardrop, udpstorm.
Probe	Probing is an attack in which the hacker scans a machine or a networking device in order to determine weaknesses or vulnerabilities that may later be exploited so as to compromise the system.	ipsweep, mscan, nmap, portsweep, saint, satan
Remote to Local (R2L)	A remote to user attack is an attack in which a user sends packets to a machine over the internet, which s/he does not have access to in order to expose the machines vulnerabilities and exploit privileges which a local user would have on the computer	ftpwrite, guesspassword, imap, multihop, named, phf, sendmail, snmpgetattack, snmpguess, warezmaster, worm, xlock, xsnoop, httptunnel.
User to Root (U2R)	User to root attacks are exploitations in which the hacker starts off on the system with a normal user account and attempts to abuse vulnerabilities in the system in order to gain super user privileges.	bufferoverflow, loadmodule, perl, rootkit, ps, sqlattack, xterm.

4.1 Model evaluation matrices

If True Positive (T_P) is the number of attacks classified rightly as attack; True Negative (T_N) is the number of normal events rightly classified normal; False Positive (F_P) is the number of normal events misclassified as attacks and False Negative (F_N) is the number of attacks misclassified as normal, we can define accuracy, recall, precision and F1 values of a model can be defined using the following equations [20].

– Accuracy: It is an indicator of the total number of correct predictions provided by the model and defined as follows:

$$\text{Accuracy} = \frac{T_P + T_N}{T_P + T_N + F_P + F_N}. \tag{5}$$

– Recall, precision and F measure: Three of the most commonly used performance measures with F measure being the harmonic mean of recall and precision measures are defined as follows:

$$\text{Recall or True positive rate} = \frac{T_P}{T_P + F_N}. \tag{6}$$

$$\text{Precision} = \frac{T_P}{T_P + F_P}. \tag{7}$$

$$\text{F Measure} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision+Recall}} \tag{8}$$

Fig. 4 showed the class-wise receiver operating characteristic (ROC) for our proposed model. The model produced highly accurate predictions for the classes of the designed NIDS. We obtained a true positive rate (TPR) of above 99% for normal connection and DoS attack category. The model benefited from comparatively higher number of instances in these two categories. The TPR for the U2R (showed in red in Fig. 4) category was 89.2%, while R2L and Probe attack types had a TPR rate of 94.3% and 98.4% respectively. While compared with the ROC curve presented in Shone et al. [1], our model achieved much higher TPR values for the under-represented intrusion types R2L and U2R.

4.2 Confusion Matrix

We presented the confusion matrix plot in Fig. 5, for our model when evaluated with the test data set. The rows correspond to the predicted class (Output Class) and the columns correspond to the true class (Target Class). The diagonal cells in the confusion matrix correspond to observations that are correctly classified (T_P and T_N's). The off-diagonal cells correspond to incorrectly classified observations

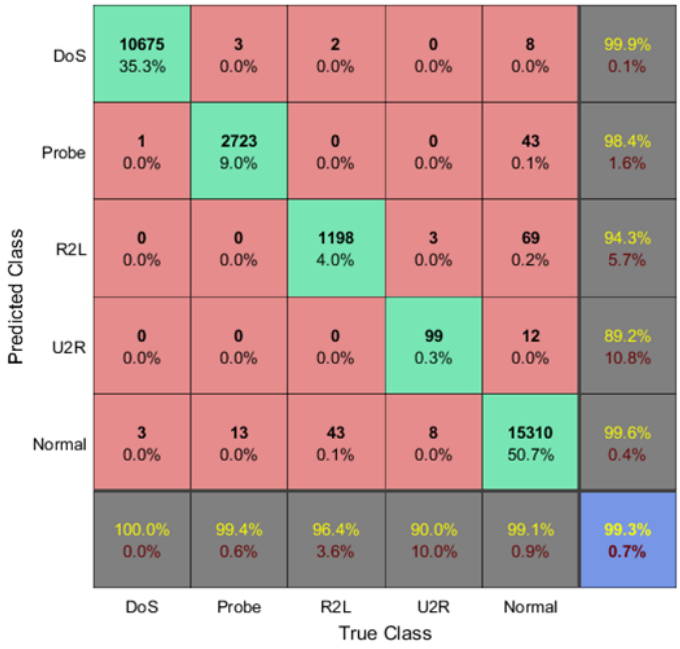

Fig. 5. Confusion matrix of the test dataset

(F_P and F_N's). Both the number of observations and the percentage of the total number of observations are shown in each cell.

The column on the far right of the plot shows the percentages of all the examples predicted to belong to each class that are correctly and incorrectly classified. These values are often called the precision and false discovery rate respectively. The row at the bottom of the plot shows the percentages of all the examples belonging to each class that are correctly and incorrectly classified. These metrics are often called the recall and false negative rate, respectively. The cell in the bottom right of the plot shows the overall accuracy of the classifier which is 99.3% in our case.

To be noted, we have utilized builtin Matlab functions *plotroc(true class, predicted class)* and *plotconfusion(true class, predicted class)* for generating Fig. 4 and Fig. 5. Both true class and predicted class variables were one-hot encoded and imported from the python interface as .mat file for plotting purposes.

4.3 Comparison with other approaches

Our results showed an excellent performance with an overall detection accuracy of 99.3% for Probe, R2L, DoS and U2R type attacks. Table 2 summarizes the class-wise true positive rate (TPR) and overall accuracy of our proposed model with some concurrent deep learning methods presented in the literature. All previously reported models in the literature showed poor performance in detecting R2L and U2R type attacks due to lack of data. We up-sampled and restructured

Table 2. Quantitative comparison of the proposed method with other deep learning techniques for NSL-KDD 5-class intrusion detection

Learning method	Year	DoS (%)	Probe	R2L	U2R	Normal	Overall Accuracy (%)
Self Taught Learning [21]	2016	-	-	-	-	-	75.76
Convolutional Neural Networks [14]	2017	-	-	-	-	-	81.57
Deep Belief Network (DBN) [1]	2018	87.96	72.97	0	0	95.64	80.58
CNN-3 layer[15]	2017	97.5	92.5	3.43	6.33	99.9	97.0
CNN 3 layer+LSTM [15]	2017	99.5	86.8	0.0	7.45	99.7	98.7
Deep Neural Network [13]	2016	97.7	89.8	13	39.6	95	93
Symmetric Autoencoder(S-NDAE) [1]	2018	94.58	94.67	3.82	2.70	97.73	85.42
Stacked Autoencoder [16]	2018	-	-	-	-	-	94.7
Proposed Autoencoded DNN	2018	99.9	98.4	94.3	89.2	99.6	99.3

the dataset to deal with this discrepancy and the trained model achieved above 89% accuracy for all the classes.

5 Conclusions

Cyber threats have become a prime concern for information security. NIDS is one of the security mechanisms used to guard these applications against attacks. In this research, we have applied a deep network intrusion detection model and evaluated the algorithm with the benchmark NSL-KDD dataset. Currently, the algorithm is trained offline on high performance computer. Our results showed an excellent performance with an overall detection accuracy of 99.3% for Probe, R2L, DoS and U2R type of attacks. We also presented a comparison with recent approches used in literature which showed a substantial improvement in terms of accuracy and latency with proposed autoencoded DNN. In future, we will provide extensions or modifications of the proposed algorithm for mobile and IoT security platforms as suggested in ref [22] using intelligent agents such as soft computing and advanced unsupervised clustering algorithms. Because of the availability of deep learning libraries in python such as Keras and TensorFlow Lite [23], on-device machine learning inference is now possible with low latency. Hence, future models will cover a broader range of attacks, respond in real time and update itself over time. Future extension of this work will be to improve the detection accuracy and to reduce the rate of false negative and false positive rate in attack detection to improve the systems performance. In future, we will also continue our efforts for improvements to the approach and will apply the

technique on similar but more recent and complex datasets such as Aegean Wi-Fi Intrusion Dataset (AWID) [24], UNSW-NB15dataset [25].

References

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

[2] B. Lee, S. Amaresh, C. Green, *et al.*, "Comparative study of deep learning models for network intrusion detection," *SMU Data Science Review, Article 8*, vol. 1, no. 1, 2018.

[3] *Nsl-kdd dataset.* [Online]. Available: `http://www.unb.ca/cic/datasets/nsl.html`.

[4] H. Liu, B. Lang, M. Liu, *et al.*, "Cnn and rnn based payload classification methods for attack detection," *Knowledge-Based Systems*, 2018.

[5] J. McHugh, "Testing intrusion detection systems: A critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," *ACM Trans. Inf. Syst. Secur.*, vol. 3, no. 4, pp. 262–294, Nov. 2000.

[6] Y. Mirsky, T. Doitshman, Y. Elovici, *et al.*, "Kitsune: An ensemble of autoencoders for online network intrusion detection," *CoRR*, vol. 1802.09089, 2018.

[7] M. A. Salama, H. F. Eid, R. A. Ramadan, *et al.*, "Hybrid intelligent intrusion detection scheme," in *Soft Computing in Industrial Applications*, A. Gaspar-Cunha, R. Takahashi, G. Schaefer, *et al.*, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 293–303.

[8] R. S. Naoum, N. A. Abid, and Z. N. Al-Sultani, "An enhanced resilient backpropagation artificial neural network for intrusion detection system," in *International Journal of Computer Science and Network Security*, vol. 12, Mar. 2012.

[9] N. Pandeeswari and G. Kumar, "Anomaly detection system in cloud environment using fuzzy clustering based ann," *Mobile Networks and Applications*, vol. 21, no. 3, pp. 494–505, Jun. 2016.

[10] N. Gao, L. Gao, Q. Gao, *et al.*, "An intrusion detection model based on deep belief networks," in *2014 Second International Conference on Advanced Cloud and Big Data*, Nov. 2014, pp. 247–252.

[11] O. Kaynar, A. G. Yüksek, Y. Görmez, *et al.*, "Intrusion detection with autoencoder based deep learning machine," in *25th Signal Processing and Communications Applications Conference (SIU)*, May 2017, pp. 1–4.

[12] Y. Li, R. Ma, and R. Jiao, "A hybrid malicious code detection method based on deep learning," 2015.

[13] 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)*, Sep. 2016, pp. 1–8.

[14] Z. Li, Z. Qin, K. Huang, *et al.*, "Intrusion detection using convolutional neural networks for representation learning," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, *et al.*, Eds., Springer International Publishing, 2017, pp. 858–866.

[15] 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)*, Sep. 2017, pp. 1222–1228.

[16] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," in *2018 18th International Conference on Advanced Communication Technology*, Feb. 2018, pp. 1–6.

[17] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[18] M. Tavallaee, E. Bagheri, W. Lu, *et al.*, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, pp. 1–6.

[19] F. Chollet. (2013). Keras: The python deep learning library, [Online]. Available: `https://keras.io/`.

[20] M. Sokolova and G. Lapalme, "A systematic analysis of performance measures for classification tasks," *Information Processing & Management*, vol. 45, no. 4, pp. 427–437, 2009.

[21] A. Javaid, Q. Niyaz, W. Sun, *et al.*, "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)*, ser. BICT'15, New York City, United States: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.

[22] J. Abawajy, S. Huda, S. Sharmeen, *et al.*, "Identifying cyber threats to mobile-iot applications in edge computing paradigm," *Future Generation Computer Systems*, 2018.

[23] *Tensorflow lite: A new mobile-specific library*, 2017. [Online]. Available: `https://www.tensorflow.org/mobile/tflite/`.

[24] *Awid dataset - wireless security datasets project*, 2014. [Online]. Available: `http://icsdweb.aegean.gr/awid/`.

[25] N. Moustafa and J. Slay, "Unsw-nb15: A comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, Nov. 2015, pp. 1–6.