

2018

## Comparative Study of Deep Learning Models for Network Intrusion Detection

Brian Lee

*Southern Methodist University*, [brianlee@smu.edu](mailto:brianlee@smu.edu)

Sandhya Amaresh

*Southern Methodist University*, [samaresh@mail.smu.edu](mailto:samaresh@mail.smu.edu)


Clifford Green

*Southern Methodist University*, [cliffordg@mail.smu.edu](mailto:cliffordg@mail.smu.edu)

Daniel Engels

*Southern Methodist University*, [dwe@smu.edu](mailto:dwe@smu.edu)

Follow this and additional works at: <https://scholar.smu.edu/datasciencereview>

 Part of the [Artificial Intelligence and Robotics Commons](#), [Digital Communications and Networking Commons](#), [Information Security Commons](#), [Other Computer Engineering Commons](#), [Other Computer Sciences Commons](#), and the [Programming Languages and Compilers Commons](#)

---

### Recommended Citation

Lee, Brian; Amaresh, Sandhya; Green, Clifford; and Engels, Daniel (2018) "Comparative Study of Deep Learning Models for Network Intrusion Detection," *SMU Data Science Review*: Vol. 1 : No. 1 , Article 8.

Available at: <https://scholar.smu.edu/datasciencereview/vol1/iss1/8>

This Article is brought to you for free and open access by SMU Scholar. It has been accepted for inclusion in SMU Data Science Review by an authorized administrator of SMU Scholar. For more information, please visit <http://digitalrepository.smu.edu>.

# Comparative Study of Deep Learning Models for Network Intrusion Detection

Clifford Green<sup>1</sup>, Brian Lee<sup>1</sup>, Sandhya Amaresh<sup>1</sup>, Daniel W. Engels<sup>1</sup>

<sup>1</sup> Southern Methodist University, Dallas, TX, US

**Abstract.** In this paper, we present a comparative evaluation of deep learning approaches to network intrusion detection. A Network Intrusion Detection System (NIDS) 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 (DoS) attacks, malware replication, and intruders that are operating within the system. Multiple deep learning approaches have been proposed for intrusion detection systems. We evaluate three models, a vanilla deep neural net (DNN), self-taught learning (STL) approach, and Recurrent Neural Network (RNN) based Long Short Term Memory (LSTM) on their accuracy and precision. Their performance is evaluated using the network intrusion dataset provided by Knowledge Discovery in Databases (KDD). This dataset was used for the third international Knowledge Discovery and Data Mining Tools competition held in conjunction with KDD Cup 1999. The results were then compared to a baseline shallow algorithm that uses multinomial logistic regression to evaluate if deep learning models perform better on this dataset.

## 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 [1]. As technology becomes more and more integrated, the challenge to keep the systems safe and away from vulnerability attacks increases. Over the years we have seen an increase in hacks in banking systems, healthcare systems and many Internet of Things (IoT) devices. 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 [2].

A network intrusion detection system (NIDS) is a software application that monitors the network traffic for malicious activity. One popular strategy is to monitor a network's activity for anomalies, or anything that deviates from normal network

behavior. Anomaly detection creates models of normal behavior for networks and other devices and then looks for deviations from those patterns of behavior at a much faster pace. Machine learning is used to build anomaly detection models and there are two approaches shallow learning and deep learning. Shallow learners mostly depend on the features used for creating the prediction model. On the other hand, deep learners have the potential to extract better representations from the raw data to create much better models. Deep learners can learn better because they are composed of the multiple hidden layers. At each layer the model can extract a better representation from the feature set when compared to shallow learners who don't have hidden layers.

In this paper, we evaluate three deep learning models that use general neural net, self-taught learning, and persistence. The latter two models we build are based on Autoencoder and Long Short Term Memory (LSTM). For this research, we use the KDD Cup 1999 Dataset for our deep learning models and compare them to soft-max regression (SMR) results performed on the NSL-KDD dataset. Soft-max regression performed yielded an accuracy of 75.23%, recall of 63.73%, and an f-measure of 75.46% [3].

## 2 Intrusion Detection

The growth of the Internet and data traffic exhibited a number of problems in regards to security management. As the Internet was not developed with security in mind, the increase in the number of users around the world had introduced the need to incorporate access controls. Intruders have gotten creative in their methods to infiltrate or disrupt network traffic. They continue to adapt to prevention mechanisms in place and continue to find ways to exploit the systems that are in place to prevent these intrusions from occurring. First designed as a rule based system in 1987, Dorothy E. Denning and Peter Neumann were the first to pioneer the Intrusion Detection Expert System (IDES) using statistical models to achieve detection of anomalies [4]. Since then, methods of attack and prevention have adapted to utilizing different mediums as those innovations continue to release new methods of connection, thus opening the window to increased vulnerabilities that have yet to be discovered [5].

Different configurations, or combinations thereof, can be used to detect a variety of known attacks. The true advantage of NIDS is the system will classify analyzed network traffic to determine if traffic or activity is normal end-user activity or malicious activity. Common methods of attack are detectable by a NIDS. In general NIDS can be configured to two different models on the host or network. The first is to detect anomalies. The detection of anomalies are achieved by establishing a baseline of normal behaviors and flagging behaviors that deviate from that baseline. The second configuration relies on the comparison of known unwanted behaviors or misuse detection [6].

Attacks can come in many forms. Several behavioral examples that would trigger a flag for an anomaly can be port scans coming from one host on a network across an entire subnet, download file count/size in a shared network folder, multiple USB file transfers, etc. NIDS can be configured to account for many behavioral examples.

More targeted configurations can account for known signatures for malware transferred across the network compared to a database containing hashes for the malware. NIDS in this form can be handled on a host-based solution. More notably, and often reported in the news, DDOS attacks can use similar configurations to block the overwhelming connection requests. This is achieved via policy in the event that the system can compare against known IP addresses; however, can also be configured to detect unknown requests that exhibit the same pattern as a DDOS attack. In all events, a response is necessary whether passive or active. In the event of behavioral triggers, a passive response, or a flag can notify a security administrator of potential compromise to intellectual assets if an employee was to download all files from a file share. The example of a DDOS attack; however, would require blocking incoming traffic requests from the detected IP address to prevent the requests from impacting the availability of a system. In all accounts, NIDS can be a powerful solution to mitigating for policing the massive amounts of data that can travel across networks, but does not replace the need for human intervention when further analysis is required to identify new threats or false positive detections [7].

**Table 1.** Definitions of attack types in the KDD Cup 1999 Dataset [8].

Attack Type	Description
DoS	A DoS attack is a type of attack in which the hacker makes a computing or memory resources too busy or too full to serve legitimate networking requests and hence denying users access to a machine.
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.
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.
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.

### 3 KDD and NSL-KDD Dataset

For our work, we use both the KDD and NSL-KDD dataset to see the difference in performance. The KDD Cup dataset was prepared using the network traffic captured by 1998 DARPA IDS evaluation program. The network traffic includes normal and different kinds of attack traffic, such as DoS, Probing, user-to-root (U2R), and root-to-local (R2L). The network traffic for training was collected for seven weeks followed by two weeks of traffic collection for testing in raw tcpdump format. The test data contains many attacks that were not injected during the training data

collection phase to make the intrusion detection task realistic. It is believed that most of the novel attacks can be derived from the known attacks. Finally, the training and test data were processed into the datasets of five million and two million TCP/IP connection records, respectively.

The KDD Cup dataset has been widely used as a benchmark dataset for many years in the evaluation of NIDS. One of the major drawback with the dataset is that it contains an enormous amount of redundant records both in the training and test data. It was observed that almost 78% and 75% records are redundant in the training and test dataset, respectively [9]. This redundancy makes the learning algorithms biased towards the frequent attack records and leads to poor classification results for the infrequent, but harmful records. The training and test data were classified with the minimum accuracy of 98% and 86% respectively using a very simple machine learning algorithm. It made the comparison task difficult for various IDSs based on different learning algorithms.

NSL-KDD was proposed to overcome the limitation of KDD Cup dataset. The dataset is derived from the KDD Cup dataset. It improved the previous dataset in two ways. First, it eliminated all the redundant records from the training and test data. Second, it partitioned all the records in the KDD Cup dataset into various difficulty levels based on the number of learning algorithms that can correctly classify the records. Further, it selected the records by random sampling of the distinct records from different difficulty levels in a fraction that is inversely proportional to their fractions in the distinct records. Each record in the NSL-KDD dataset consists of 41 features and is labeled with either normal or a kind of attack. These features include basic features derived directly from a TCP/IP connection, traffic features accumulated in a window interval, either time, e.g. two seconds, or many connections, and content features extracted from the application layer data of connections. When comparing the accuracy of our model against the KDD and NSL-KDD dataset, KDD fared better yielding a higher accuracy. Though the NSL-KDD dataset has been cleaned and optimized for machine learning purposes, we discover that reduction in dimensionality in our deep learning models have a substantial impact on the accuracy when executing against the test set of the original KDD dataset. In addition, there are significant differences in the sizes of the training sets for the two datasets. KDD dataset contains 370,515 records while the NSL-KDD dataset contains 125,974. Effectively, the deep learning model has alleviated the requirement of a manual data step.

## 4 Deep Learning Models

Deep learning was inspired by the structure and depth of human brain. Because of the multiple levels of abstraction, the network learns to map the input features to the output. The process of learning does not depend on human-crafted features. Given a set of conditions, the machine can use a series of mathematical methods to determine if a classification is accurate based on the likelihood of error. Within the realm of deep learning, we focus on deep networks where the classification training is

conducted by training with many layers in hierarchical networks with unsupervised learning. Deep network intrusion detection systems can be classified based on how the architectures and techniques are being used.

In this section, we mention the models used for analysis. The first model is a vanilla deep neural net classifier, which can be thought of as stacked logistic regressors. The second is the self-taught learning model using autoencoder and the third is Recurrent Neural Network will be using Long Short Term Memory. To measure the performance of these models we use the metrics mentioned in Table 2.

**Table 2.** Model Evaluation Metrics.

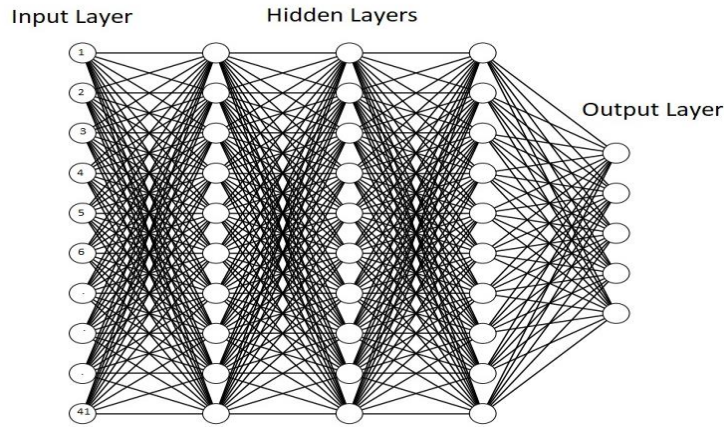
Attack Type	Description
Accuracy	Defined as the percentage of correctly classified records over the total number of records.
Precision (P)	Defined as the % ratio of the number of true positives (TP) records divided by the number of true positives (TP) and false positives (FP) classified records. $P = TP / (TP + FP) \times 100\%$
Recall (R)	Defined as the % ratio of number of true positives records divided by the number of true positives and false negatives (FN) classified records. $R = TP / (TP + FN) \times 100\%$
F-Measure (F)	Defined as the harmonic mean of precision and recall and represents a balance between them. $F = 2.P.R / (P+R)$

## 4.1 Deep Neural Net

A Deep Neural Network is essentially a multilayer perceptron, which was initially developed by stacking linear classifiers. This is the most basic type of Deep Neural Network that exists. The model is fed inputs, inputs get multiplied by weights and the passed into an activation function. In a Deep Neural Network, this process occurs over multiple layers. The model uses backpropagation to adjust weights and increase accuracy. Any model than contains 3 or more layers is considered a deep network.

### 4.1.1 Model Setup

Prior to training the model the data were prepared by converting categorical features to numeric values. The data were normalized to reduce training time and increase performance. The final dimension of the dataset were 41 different features with 5 different predicted classes.



**Fig. 1.** A Deep Neural Network with 3 hidden layers.

#### 4.1.2 Results

The deep neural network attained an accuracy of 66%, the classification of each attack type is shown below in Figure 2. The model was able to classify DoS and probe attacks well but had little success in correctly classifying normal non-threatening requests and U2R attacks. The accuracy is a lot lower than expected from a deep network.

	precision	recall	f1-score	support
DoS	0.83	0.62	0.71	4342
Probe	0.80	0.68	0.74	2402
R2L	0.67	0.00	0.00	2753
U2R	0.00	0.00	0.00	200
normal	0.23	0.69	0.34	2152
avg / total	0.66	0.49	0.47	11849

**Fig. 2.** Deep neural net results.

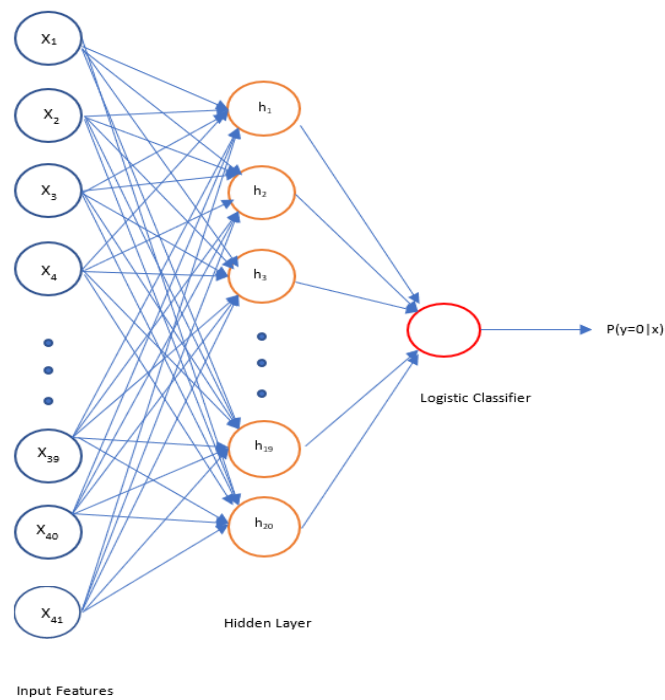
#### 4.2 Self-Taught Learning Approach

Self-taught learning (STL) is a deep learning approach that consists of two stages for classification. The first stage is Unsupervised Feature learning that consists of learning a good feature representation from a large collection of unlabeled data. This

stage is implemented using a sparse Autoencoder. A sparse autoencoder is a neural network that consists of input, hidden and output layers. The input and output layers contain equal  $N$  nodes, while the hidden layer contains  $K$  nodes. The output from the autoencoder is then passed through a soft-max regression (SMR) for the classification task.

#### 4.2.1 Model Setup

Before using the training dataset, we first convert the categorical features to numeric values. We then perform a min-max normalization on this feature vector. The labels are one hot encoded. Therefore, the input dimension is 41 and output dimension is 5 (4 attacks and 1 normal). We pass the feature vector through a two-layer stacked autoencoder, the first autoencoder has a hidden layer of 20 and the second layer has a hidden layer of 10. The output from the encoder of the second layer is then passed through a soft max regressor to classify the input to one of the 5 labels.



**Fig. 3.** Autoencoder dimensionality reduction and features input to a logistic classifier.



#### 4.2.2 Results

The STL approach results in an accuracy of 98.9% with the following break up by each attack type. We observe that since the representation of R2L and U2R type of attacks are low the precision and recall of these attack types are lesser when compared to the other attack types. We see that the STL has learned a good representation of the feature set to be able to predict with a high degree of accuracy.

	precision	recall	f1-score	support
DoS	1.00	0.99	0.99	97865
Probe	0.78	0.59	0.67	1027
R2L	0.51	0.11	0.19	281
U2R	0.00	0.00	0.00	13
normal	0.95	0.98	0.96	24320
avg / total	0.98	0.98	0.98	123506

**Fig. 4.** Self-taught learning (autoencoder) results.

#### 4.3 Recurrent Neural Network

Recurrent neural networks are a class of Artificial Neural network. They take as their input not only the current input instance but also what they have perceived previously in time. This means that they also have an additional memory input. The decision a RNN takes at time  $t-1$  influences the decision it takes at time  $t$ . So, the recurrent neural networks have two sources of input – the present and the recent past, which combine to determine how the RNN will respond to the new data. This feedback loop is main difference between RNNs and the feed forward neural network. One of the short comings of a RNN was the vanishing gradient problem. This happens when the gradient is very small, and hence the weights cannot be changed. This would prevent the neural net from training further. The Long Short-Term Memory networks (LSTM) – are a special kind of RNN, which eliminates the vanishing gradient issue, as they can learn long term dependencies easily. Normal RNNs take in their previous hidden state and the current input state to output a new hidden state. The LSTM does the same, except it also takes an old cell state.

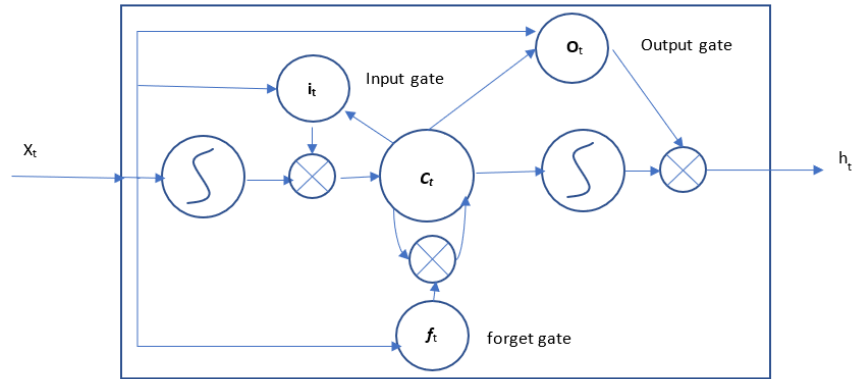


Fig. 5. Long Short Term Memory (LSTM) cell.

#### 4.3.1 Model Setup

Similar to the previous model setup we first convert the categorical features to numeric values. We then perform a min-max normalization on this feature vector. The labels are one hot encoded. Therefore, the input dimension is 41 and output dimension is 5 (4 attacks and 1 normal). And we apply LSTM architecture to the hidden layer. The time step size, batch size, and epochs are 100, 50, 5 respectively. We use softmax for the output layer and stochastic gradient descent (SGD) for an optimizer. We use a learning rate of 0.01 and hidden layer of 80.

#### 4.3.2 Results

The LSTM model results in an accuracy of 79.2% with the following break up by each attack type. We observe that this model is unable to predict attacks other than DoS. This may be due to the training data having a higher distribution of DoS instances and may need further tuning of our model.

	precision	recall	f1-score	support
DoS	0.79	1.00	0.88	97780
Probe	0.00	0.00	0.00	1027
R2L	0.00	0.00	0.00	281
U2R	0.00	0.00	0.00	13
normal	0.00	0.00	0.00	24304
avg / total	0.63	0.79	0.70	123405

Fig. 6. Long Short Term Memory (LSTM) model results.

#### 4.4 Results Analysis

We can see the comparison of the performance metrics in figure 7. The comparison is made between the results of our three models: DNN, RNN, and Autoencoder deep learning algorithms. Overall, Autoencoder had performed with the highest precision, recall, and f1-score between distinguishing DoS type attacks with normal network traffic. All models were unable to detect U2R type attacks due to lack of data to successfully perform classification.

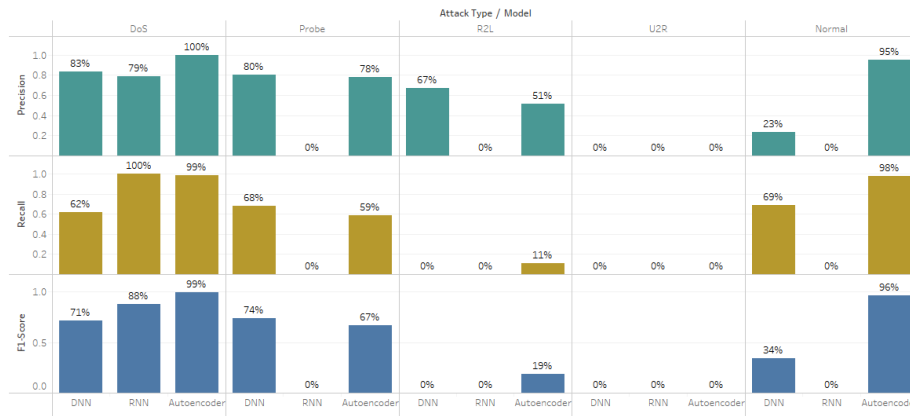


Fig. 7. Performance metric result comparison between DNN, RNN, and Autoencoder.

## 5 Ethical Considerations

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. It can help optimize the use of resources, whether human or machine; however, it cannot be the sole solution in the attempt to mitigate the risk of intrusion [4]. The solution to the bigger picture of cyber security is that different solutions need to exist at every layer of the network to effectively say that a network is "secure." Machine learning is just part of a bigger picture, albeit a very large part. To give an idea of how much, imagine how many people it would take manually analyze 1 million records. The capabilities of implementing a NIDS using deep learning would greatly alleviate load currently placed on resources throughout the detection process. This does not come without inherent risks. With the capability for a single individual or a team of individuals to draw conclusions or inferences from these massive amounts of data, upholding the highest level of integrity is paramount in the continued development in

the field of data science. This alludes us to the ACM Code of Ethics and Professional Conduct [9]. Speaking from a general standpoint, it is the ethical responsibility of the professional to “contribute to society and human well-being.” However, there are many instances where research results can inadvertently cause harm to an individual or a group of individuals. Worse yet, instances where these activities could be deliberate.

Examples of the mishandling of data can be found in across different industries from the accidental termination of employees to the intentional manipulation of earnings reports [10][11]. To align these incidences to the topic of NIDS, a scientist with access to the data, ethically handled, will disclose any nuances that are attributed to the dataset throughout their involvement in the research and report. For example, a training set contains false positives of IP addresses thought to be from a questionable location, but the scientist knows that the addresses are those of satellite offices within the companies known network. Though, if the scientist knowingly reports the results of their research with known skewed results, but does not mention the contained biases or otherwise on behalf of the entity of whom they are doing the research for, would be unethical. Even more unethical, if the scientist was to manipulate the training data for this specific satellite office due to some personal vendetta against a particular person only to claim later, that it was an oversight.

In relation to the former example, if these results are to be used to make decisions, using a dataset with known biases would fall into the realm of unethical actions. The ethical response, in this case, would be to conclude that due to the known biases, any result that the study yields would be considered unactionable. It is for these reasons that ethics continues to play a pivotal role in the continued development of deep learning algorithms.

## 6 Conclusion

We observe that the autoencoder is able to classify the attack types with an accuracy of 98.9%. In contrast, the Long Short Term Memory (LSTM) model yielded a 79.2%. Further tuning of hyperparameters is likely required to improve the accuracy of the LSTM model. Because the prediction of these models has a reliance on the training set, the class imbalance could be the cause of lower accuracy.

The self-taught learning model, as a result of dimensionality reduction, reduces the number a features to 10 in the autoencoder. The result is greater accuracy in comparison to the SMR results performed on the cleaned NSL-KDD dataset, which yielded a much lower 75.23% taking into account all 41 features in the original dataset. We can conclude that the autoencoder deep learning algorithm is a good model for NIDS.

In application, the STL model could be implemented in an environment where data is unclean. Though, an important consideration is to recognize that the best practice for applying any model, would be to ensure that the data is clean. The results of the analysis between the deep learning models suggests that the use of deep learning in NIDS would be a suitable solution to improving detection accuracy on unclean data; however, building an environment that is specifically designed for this purpose would

go a long way to further improve and could greatly impact the decision on which model would work best in a given environment.

## References

1. Evans, D.: The Internet of Things How the Next Evolution of the Internet Is Changing Everything (2011)
2. Gers, F.: Long Short-Term Memory in Recurrent Neural Networks (2001)
3. Niyaz, Q., Sun, W., Javaid, A.Y., Alam M.: A Deep Learning Approach for Network Intrusion Detection System (2015)
4. Amad, B., Jian, W., Hassan, B., Rehmatullah, S.: Hybrid Intrusion Detection Method to Increase Anomaly Detection by Using Data Mining Techniques. International Journal of Database Theory and Application (2017) 231-240
5. W. Lee and B. Rotoloni, Emerging Cyber Threats, Trends & Technologies Report. Georgia Tech Institute for Information Security & Privacy (2004)
6. Bruneau, G.: The History and Evolution of Intrusion Detection. <https://www.sans.org/reading-room/whitepapers/detection/history-evolution-intrusion-detection-344> (2001)
7. Bunel, P.: An introduction to Intrusion Detection Systems. London, England: SANS Institute. <https://www.giac.org/paper/gsec/4227/introduction-intrusion-detection-systems/106775> (2017)
8. Paliwal, S., Gupta, R.: Denial-of-Service, Probing & Remote to User (R2L) Attack Detection using Genetic Algorithm, 60th ed. International Journal of Computer Applications (2017)
9. ACM Code of Ethics and Professional Conduct, Acm.org (2017)
10. Chew, J.: Marissa Mayer Just Fired Dozens of Yahoo Employees By Accident. Fortune (2016)
11. Trainer, D.: Four Reasons Executives Manipulate Earnings. Forbes (2017)
12. Dong B., Wang X.: Comparison Deep Learning Method to Traditional Methods Using for Network Intrusion Detection (2016)
13. Kruegel, C., Mutz, D., Robertson, W., Valeur, F.: Bayesian event classification for intrusion detection. Proc. 19th Annual Computer Security Applications Conference (2003) 14-23
14. Sinclair, C., Pierce, L., Matzner, S.: An application of machine learning to network intrusion detection. Proc. 15th Annual Computer Security Applications (1999) 371-377
15. Zhang, J., Zulkernine, M.: A hybrid network intrusion detection technique using random forests. Proc. First International Conference on Availability, Reliability and Security (ARES'06), April (2006) 8-16
16. Yang, J., Deng, J., Li, S., Hao, Y.: Improved traffic detection with support vector machine based on restricted Boltzmann machine. Soft Computing, vol. 19 (2015) 1-12
17. M. Lincoln Labs. DARPA Intrusion Detection Evaluation. <http://www.ll.mit.edu/IST/ideval> (1999)
18. Hinton, G.E.: The "wake-sleep" algorithm for unsupervised neural networks. Science 268.5214 (1995) 1158
19. Hinton, G.E., Osindero S., Teh Y.W.: A fast learning algorithm for deep belief nets. Neural computation 18.7 (2006) 1527-1554
20. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. Advances in neural information processing systems (2007)
21. Torres, P., Catania, C., Garcia, S., Garino, C.: An Analysis of Recurrent Neural Networks for Botnet Detection Behavior (2016)
22. Hayun, L.: Demystifying Machine Learning ('Artificial Intelligence') Use in Endpoint Security Products. Proc. Palo Alto Networks Ignite 2017 Conference (2017)

23. IT Industry Outlook 2017. CompTIA. <https://www.comptia.org/resources/it-industry-trends-analysis-2017> (2017)
24. Building digital trust: The role of data ethics in the digital age. Accenture (2017)