International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 1 Volume No.6 January – 2022 DOI: 10.46647/ijetms.2022.v06i01.006
ISSN: 2581-4621

Machine Learning Algorithms to Classify Network Attacks

S.Bhaskara Naik[1], R.Sasikala[2]

[1]Lecturer in Computer Science, S.V.B Govt. Degree College, Koilakuntla, Andhra Pradesh, India
[2]Assistant Professor, Dept. Of CSE, Karpagam College of Engineering,Coimbatore, Tamilnadu,India
[1]baskaranaik999@gmail.com
[2]sasikala.r@kce.ac.in

Abstract— **An intrusion detection system (IDS) identifies whether the network traffic behavior is normal or abnormal or identifies the attack types. Recently, deep learning has emerged as a successful approach in IDSs, having a high accuracy rate with its distinctive learning mechanism. The most common operational network intrusion detection systems are signature based systems. These systems consist of a database of attack signatures. Human experts produce the attack signatures by manually analyzing the attack data. The monitored network traffic is matched against this database to detect malicious activities. Producing attack signatures is a time consuming and manually intensive task.**
Recently, machine learning techniques have been applied to build predictive models for the detection of network attacks. Unlike signature based methods, which need manual analysis by human experts to extract attack patterns, machine learning algorithms are able to automatically extract similarities and patterns in the network data.
We found that using LSTM-RNN classifiers with the optimal feature set improves intrusion detection. The performance of the IDS was analyzed by calculating the accuracy, recall, precision, f-score, and confusion matrix. The NSL-KDD dataset was used to analyze the performances of the classifiers. An LSTM-RNN was used to classify the NSL-KDD datasets into binary (normal and abnormal) and multi-class (Normal, DoS, Probing, U2R, and R2L) sets. The results indicate that applying the GA increases the classification accuracy of LSTM-RNN in both binary and multi-class classification. The results of the LSTM-RNN classifier were also compared with the results using a support vector machine (SVM) and random forest (RF). For multi-class classification, the classification accuracy of LSTM-RNN with the GA model is much higher than SVM and RF. For binary classification, the classification accuracy of LSTM-RNN is similar to that of RF and higher than that of SVM.

Keywords—**Machine Learning, Algorithms, Network Attacks, Genetic Algorithms, LSTM-RNN.**

I. Introduction

The rapidly growing progress in Internet-based technology has brought tremendous benefits to our society. Communications and other services that the Internet provides have transformed our lives in many ways. The Internet has opened up a whole new world of possibilities to access the information. Students and researchers do not need to go to the libraries to collect the information they need anymore. Nowadays, the information is just a few clicks away from one's computer web browser. Social networking sites have eliminated geographic distance and made it easier to be in contact with family and friends. Online services, such as online shopping, online banking, and online learning, have made all these activities more convenient to do.

While the Internet has made our lives much more convenient, its vulnerabilities and the amount of information communicating over it generate opportunities for adversaries to perform malicious activities within its infrastructure. Any host connected to the public Internet or even a private network is under constant threat from potential attacks. A lot of threats are created every day by individuals and organizations to attack computer networks to steal private information and data. This information can be very critical and sensitive, such as social security numbers or bank account information. This has created the need for security technologies to secure users' information and provide reliable computer network environments. Network security has become a very important factor for the companies and organizations to consider. In that regard, intrusion detection plays an important role in the detection of attacks and with securing computer networks. Intrusion Detection Systems (IDS) monitor and analyze network systems to detect malicious activities. Even though users benefit from the use of IDS technology, more is needed to detect better obfuscated or more complex attack patterns.

II. Machine Learning for the Detection of Network Attacks

Machine learning is a subfield of computer science, which uses pattern recognition and artificial intelligence methods to group and extract behaviours and entities from the data. These previously known patterns and relationships trained by machine learning algorithms can be used to do prediction tasks on new data. With today's technology, machine learning algorithms touch our everyday life by being used in a wide range of applications. Examples from common domains, which machine learning algorithms are extensively used, include product recommendations systems, such as the ones used by Amazon and Netflix; natural language processing; spam detection, image recognition and fraud detection.

The most common operational network intrusion detection systems are signature based systems. These systems consist of a database of attack signatures. Human experts produce the attack signatures by manually analysing the attack data. The monitored network traffic is matched against this database to detect malicious activities. Producing attack signatures is a time consuming and manually intensive task.

Recently, machine learning techniques have been applied to build predictive models for the detection of network attacks. Unlike signature based methods, which need manual analysis by human experts to extract attack patterns, machine learning algorithms are able to automatically extract similarities and patterns in the network data. With more data being produced than the human brain has the capacity to monitor, machine learning analysis provides results that even an army of analyst experts would be unable to accomplish. With machine learning affecting a lot of aspects in our everyday life, it is necessary to study the usage of its interdisciplinary capabilities in the detection of computer network attacks. Machine learning algorithms can be applied on network data to extract patterns and similarities, which distinguish between normal and attack instances. These trained patterns can be used to build intrusion detection systems for the detection of network attacks. With the bulk of the work being carried out by machine learning, the cyber security experts can become more productive by focusing on analytical results from machine learning models in order to get more insight about the current and future threats.

III. Background

A. Deep Learning Architecture

Deep learning is one of the machine learning methods that implements artificial neural networks. A deep learning network is a multi-layer neural network. Deep learning networks include deep neural networks (DNN), convolutional neural networks (CNN), recurrent neural networks (RNN), deep belief networks (DBN), and others. In this section, we will describe the architecture of RNN and long short-term memory (LSTM).

B. Recurrent Neural Network (RNN)

RNN is a type of artificial neural network (ANN), wherein the connection between the nodes resembles the neurons of a human brain. Neural network connections can transmit signals to other neurons/nodes like synapses in a biological brain. The artificial neuron then processes the received signal and transmits it to the other connected neurons/nodes. Neurons and connections typically have weights to adjust the learning process. The weight can vary to adjust the strength of the signal as the signal travels from the input layers to the output layers. An ANN contains hidden layers between the input and output layers. RNN should have at least three hidden layers. The basic architecture of RNNs includes input units, output units, and hidden units, with the hidden units performing all the calculations by weight adjustment to produce the outputs. The RNN model has a one-way flow of information from the input units to the hidden units and a directional loop that compares the error of this hidden layer to that of the previous hidden layer, and adjusts the weights between the hidden layers. Figure 1 represents a simple RNN architecture with two hidden layers.

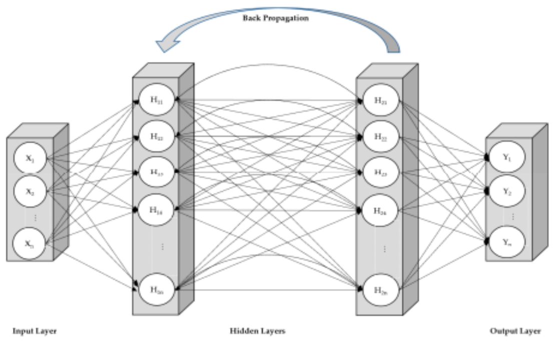

Fig 1: A Simple RNN

An RNN is an extension of traditional feed-forward neural networks (FFNNs). In FFNNs, the information moves in only the forward direction; i.e., from the input nodes, through the hidden nodes, to the output nodes; there are no cycles or loops in the network. Hidden layers are optional in traditional FFNNs. We assume an input vector sequence, a hidden vector sequence, and an output vector sequence denoted by X, H, and Y, respectively.

RNN uses gradient-based methods to learn time sequences: back-propagation through time (BPTT) or real-time recurrent learning (RTRL). In BPTT, the network is unfolded into a multilayer FFNN in which each time a sequence is processed to construct the FFNN; firstly the training data is used to train the model and then the output error gradient is saved for each time step. BPTT uses the standard back propagation algorithm to train each FFNN, and it updates the weights using the sum of the gradients obtained for weights in all layers of the network. RTRL is an online learning algorithm, where the error gradient is computed, and weights are updated for each time step in a forward propagation manner. It computes the gradients of the internal and output nodes with respect to all the weights of the network. Standard RNNs are not able to establish more than 5–10 time steps. A vanishing gradient problem may arise in RNN when gradient-based learning methods are used for updating the weights. Weights receive an updated proportion of the partial derivative of the error function in each training iteration. In some cases, the gradient will be very small. These error signals may either blow-up or vanish, which prevents the weight from changing value. These vanishing error signals may cause the weights to fluctuate. With a vanishing error, learning takes an unacceptable amount of time or does not work at all. In [25], a detailed theoretical analysis and solution of that problem with long-term dependencies is presented. RNNs can be used for supervised classification learning. RNNs are difficult to train because of vanishing and exploding gradients. The problems of vanishing and exploding gradients arise due to improperly assigned weights (assigned to either very high or very low value). Thus, an LSTM with forget gates is often combined with an RNN to overcome these training issues [26,27]. However, RNNs are a good choice for solving time series sequence prediction problems. In [18–20], researchers showed that some implementations of LSTM-RNN provide notable performance in intrusion detection. In [21], Staudemeyer et al. used LSTM-RNN on the KDD Cup'99 datasets and showed that LSTM-RNN could learn all attack classes hidden in the training data. They also learned from their experiment that the receiver operating characteristics (ROC) curve and the AUC (area under the ROC curve) are well suited for selecting high performing networks. The ROC curve is the probability curve plotting the TPR against the FPR, where TPR is on the y-axis and FPR is on the x-axis. A ROC curve shows the performance of a classification model at all classification thresholds. AUC value measures the entire two-dimensional area under the ROC curve. AUC values range from 0 to 1. AUC = 0.0 means the prediction of the model is 100% wrong and AUC = 1.0 means the prediction is 100% correct.

C. Long Short-Term Memory

LSTM can mitigate the problem of vanishing error [19 ,21,23]. LSTM can learn how to bridge more than 1000 discrete time steps [23]. LSTM networks replace all units in the hidden layer with memory blocks. Each memory block has at least one memory cell. Figure 2 demonstrates one cell in a basic LSTM network. The memory cells activate with the regulating gates. These gates control the incoming and outgoing information flow. A forget-gate is placed between an input gate and an output gate. Forget gates can reset the state of the linear unit if the stored information is no longer needed. These gates are simple sigmoid threshold units. These activation functions range from 0 to 1.

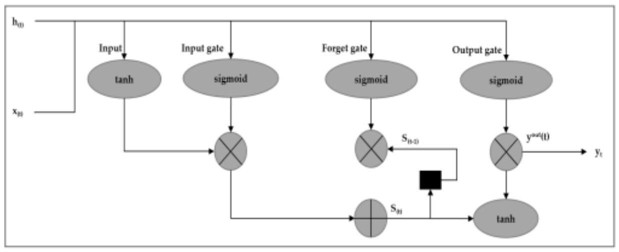

Fig 2: LSTM Model

IV. LSTM-RNN NIDS WITH A GENETIC ALGORITHM

This section describes the LSTM-RNN NIDS with the feature-selection GA. The RNN is the basic model, and a LSTM network is used to achieve a high detection rate. In this research, the training dataset was used to train the classifier, and the testing dataset was then used to measure the accuracy of the classifier. Two types of classifications were conducted: binary and multi-class. Normal and anomaly are the two classes in binary classification, whereas normal, denial of service (DoS), probe, user-to root (U2R), and remote-to-local (R2L) are the five categories detected using multi-class classification. The classification metrics considered in this research are accuracy, precision, recall, f-score, true positive rate, and false-positive rate. The confusion matrix was calculated to show the records of true positive, true negative, false positive, and false negative records achieved in each model. The LSTM-RNN models were designed with 5, 10, 20, 40, 60, 80, and 100 neurons in the hidden layers. Python 3.7 with the scikit-learn, TensorFlow, and Keras packages was used to develop the programs along with other Python open-source libraries.

A. LSTM-RNN Classifier with GA Architecture

Figure 3 shows the following steps performed for classifying network attacks using the LSTM-RNN classifier along with the GA.

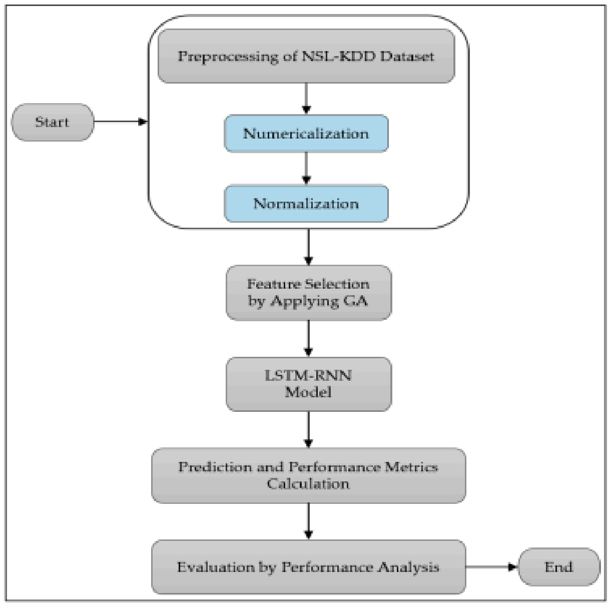
Start
Preprocessing of NSL-KDD Dataset
Numericalization
Normalization
Feature Selection by Applying GA
LSTM-RNN Model
Prediction and Performance Metrics Calculation
Evaluation by Performance Analysis
End

Fig 3: LSTM-RNN Classifier with CA

Step 1: Data Preprocessing
Numericalization: Our training and testing datasets include 38 numeric features and three nonnumeric features. We must make all the features numeric to apply them the model, so we converted the non-numeric features, protocol-type, service, and flag, into numerals. There were three values in the protocol type, 70 values in service and 11 values in flag features. In transforming all non-numeric features, we mapped 41 features into 122 (= 38 + 3 + 70 + 11) features [15]. To transform the categorical features into binary features, we first transformed the features into integers using LabelEncoder, a scikit-learn package. Then the integers were passed to One-HotEncoder to transform them into binary features. The One-Hot-Encoder is a matrix of integers, which represent different sub-categories of categorical features. The output is a sparse matrix. Each column of the output matrix corresponds to one possible value of one feature. The values of the input features have the range [0, n]. Normalization: Some features in the feature space, such as duration, src_bytes, and dst_bytes, have a very large difference between the maximum and minimum values. We applied a logarithmic scaling method to map to the range [0,1][15]. Normalization is not required for every dataset, but a dataset like NSL-KDD, where the features have different ranges of values, will require normalizing the values. Normalization changes the values of a dataset to a common scale. If the features do not have a similar range of values, then the gradient descents may take too long to converge. By normalizing, we can assure that the gradient descents can converge more quickly. An added benefit is that normalizing data can also increase the accuracy of a classifier.

Step 2: Feature Selection
Feature selection from the dataset is the process of finding the most relevant feature-set for the future predictive model implementation. This technique was used to identify and remove the unneeded, redundant, and irrelevant features that do not contribute or decrease the accuracy of the future predictive model implementation for the classification. Optimal feature selection is one of the most important tasks in developing an excellent NIDS because some features can bias the classifier to identify a particular class, and that may increase the amount of misclassification. To minimize the misclassification rate, to minimize the training time, and to maximize the accuracy of the classifier, we have applied a GA in our experiment. Using a GA, we have obtained an optimal subset of 99 features from the original 122 features. After applying the GA, insignificant data are removed from the original features set. The GA selects only a subset of relevant features. Accuracy is used as the fitness function in this research. We have calculated the maximum, minimum and average accuracy of the training dataset according to their labels. The GA provided a set of solutions by performing recombination and mutation of the features and selected the features with maximum, minimum, and average accuracy. The subset of features that gives the maximum accuracy was selected as the optimal feature set. We then prepared our training and testing datasets with this optimal feature set. The flow-chart of the GA implementation is shown in Figure 4.

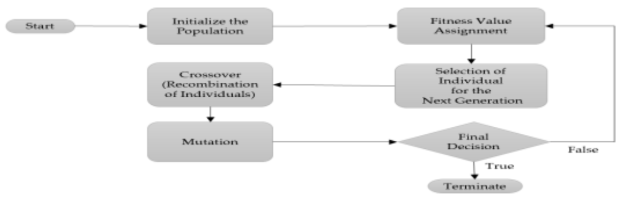
Start
Initialize the Population
Fitness Value Assignment
Crossover (Recombination of Individuals)
Selection of Individual for the Next Generation
Mutation
Final Decision
False
True
Terminate

Fig 4: The flowchart of genetic algorithm implementation

Step 3: LSTM-RNN Model Construction
To build the LSTM-RNN model, first we selected hyper-parameters and optimizers for both binary and multi-class classification. We determined the following hyper-parameters: batch size, the number of epochs, learning rate, dropout, and activation function.

a. Batch size is the number of training records in one forward and one backward pass.

b. An epoch means one forward and one backward pass of all the training examples.

c. Learning rate is the proportion of the weights that are updated during the training of the LSTM-RNN model. It can be chosen from the range [0.0–1.0].

d. Dropout is a regularization technique, where randomly selected neurons are ignored during training. The selected neurons are temporarily removed on the forward pass.

e. The activation function converts an input signal of a node in a neural network to an output signal, which is then used as the input of the next hidden layer and so on. To find the output of a layer, we calculated the sum of the products of inputs and their corresponding weights, applied the activation function to that sum, and then fed that output as an input to the next layer.

A suitable optimizer plays a very crucial part in classification. From various available optimizers, we selected stochastic gradient descent (SGD) for multiclass classification and the Adam optimizer for binary classification. SGD provides a lower computational cost than other gradient descent optimizers used in multiclass classification. Because other optimizers, such as the batch gradient descent and mini-batch gradient descent optimizers, use all the training samples for completing one iteration, they are computationally expensive to use. The advantage of SGD is that it uses only one batch for each iteration.

The Adam optimizer is easy to implement and computationally efficient because the decision is in binary and it requires less memory to implement.

Step 4: Prediction and performance metrics calculation

After fitting the model using the training dataset and testing dataset, we obtained the testing accuracy rate and loss value of the classifier. The testing dataset is used as a validation set to get an unbiased estimate of accuracy during the learning process. Then we used the predict_classes module of Keras to get the prediction matrix of the model; that is the classes predicted by the model. At this point, the expected classes have been defined in the original KDDTest+ dataset and the predicted classes. Next, we calculated the classification metrics such as precision, recall, f-score, true positive rate (TPR), false-positive rate (FPR), and confusion matrix from the expected and predicted classes matrix.

Step 5: Evaluation by performance analysis

We performed Step 4 for both SVM and RF classifiers and then evaluated the performances of SVM, RF, and LSTM-RNN by comparing the metrics.

V. Experimental Results

In this section, we present the classification performance of the LSTM-RNN classifier according to accuracy, precision, recall, f-score, TPR and FPR on 5, 10, 20, 40, 60, 80, and 100 neurons in the hidden layers, respectively. The classification performance is analyzed for both binary and 5-class classification. For 5-class classification, the classes include normal, DoS, probe, U2R, and R2L attacks, while for binary classification, the classes include normal and anomaly. We also compared the performance of LSTM-RNN with traditional machine learning approaches such as SVM and RF using the same mixed feature sets. We compared the performance using the 122-feature set and the 99-feature set, which were obtained with the genetic algorithm.

Table 6. Binary classification performance results using 122 features.

No. of Neurons in Hidden Layers	Accuracy		Precision	Recall	f_1-Score	TPR	FPR
	Training %	Testing %					
5	**99.88**	**96.51**	**0.97**	**0.97**	**0.97**	**0.944**	**0.007**
10	99.90	96.29	0.96	0.96	0.96	0.944	0.012
20	99.90	96.41	0.96	0.96	0.96	0.942	0.006
40	99.89	96.28	0.95	0.96	0.96	0.943	0.011
60	99.88	96.25	0.96	0.96	0.96	0.939	0.006
80	99.86	96.04	0.96	0.96	0.96	0.936	0.007
100	99.85	95.94	0.96	0.96	0.96	0.934	0.008

Table 7. Multi-class classification performance results using 122 features.

No. of Neurons in Hidden Layers	Accuracy		Precision	Recall	f_1-Score	TPR	FPR
	Train %	Test %					
5	96.00	85.65	0.86	0.86	0.85	0.765	0.023
10	98.00	78.13	0.80	0.78	0.73	0.630	0.019
20	98.70	76.61	0.72	0.74	0.69	0.589	0.060
40	99.89	82.42	0.87	0.82	0.81	0.697	0.008
60	**99.87**	**82.68**	**0.87**	**0.83**	**0.81**	**0.707**	**0.015**
80	99.84	81.33	0.87	0.81	0.79	0.685	0.017
100	99.85	82.06	0.85	0.82	0.81	0.689	0.006

VI. Conclusion

In this paper, we compared different classifiers on the NSL-KDD dataset for both binary and multi-class classification. We considered SVM, random forest, and the LSTM-RNN model. We have shown that our proposed model produced the highest accuracy rate of 96.51% and 99.91% for binary classification using 122 features and an optimal set of 99 features, respectively. The LSTM-RNN obtained higher accuracy than the SVM in binary classification. However, random forest was the best classifier among all in that case. However, using the 99-feature set, we were able to get testing accuracy similar to that of RF.

References

1. Denning, D.E. An intrusion-detection model. IEEE Trans. Softw. Eng. 1987, 13, 222–232. [CrossRef]
2. Peddabachigari, S.; Abraham, A.; Thomas, J. Intrusion Detection Systems Using Decision Trees and Support Vector Machines. Int. J. Appl. Sci. Comput. 2004, 11, 118–134.
3. Rai, K.; Devi, M.S.; Guleria, A. Decision Tree Based Algorithm for Intrusion Detection. Int. J. Adv. Netw. Appl. 2016, 7, 2828–2834.
4. Ingre, B.; Yadav, A.; Soni, A. K Decision Tree-Based Intrusion Detection System for NSL-KDD Dataset. In

Proceedings of the International Conference on Information and Communication Technology for Intelligent Systems, Ahmedabad, India, 25–26 March 2017. [CrossRef]
5. Farnaaz, N.; Jabbar, M.A. Random Forest Modeling for Network Intrusion Detection System. Procedia Comput. Sci. 2016, 89, 213–217. [CrossRef]
6. Alom, M.Z.; Taha, T.M. Network intrusion detection for cybersecurity using unsupervised deep learning approaches. In Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON), Dayton, OH, USA, 27–30 June 2017. [CrossRef]
7. Yuan, Y.; Huo, L.; Hogrefe, D. Two Layers Multi-class Detection Method for Network Intrusion Detection System. In Proceedings of the IEEE Symposium on Computers and Communications (ISCC), Heraklion, Greece, 3–6 July 2017. [CrossRef]
8. Gurav, R.; Junnarkar, A.A. Classifying Attacks in NIDS Using Naïve- Bayes and MLP. Int. J. Sci. Eng. Technol. Res. (IJSETR) 2015, 4, 2440–2443.
9. Tangi, S.D.; Ingale, M.D. A Survey: Importance of ANN-based NIDS in Detection of DoS Attacks. Int. J. Comput. Appl. 2013, 83. [CrossRef]
10. Szegedy, C.; Toshev, A.; Erhan, D. Deep Neural Networks for Object Detection. In Proceedings of the 26th International Conference on Neural Information Processing Systems—Volume 2; Curran Associates Inc.: Red Hook, NY, USA, 2013; pp. 2553–2561.
11. Wang, M.; Huang, Q.; Zhang, J.; Li, Z.; Pu, H.; Lei, J.; Wang, L. Deep Learning Approaches for Voice Activity Detection. In Proceedings of the International Conference on Cyber Security Intelligence and Analytics, Shenyang, China, 21–22 February 2019; pp. 816–826.
12. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi, S.A.R.; Ghogho, M. Deep Learning Approach for Network Intrusion Detection in Software-Defined Networking. In Proceedings of the 2016 International Conference on Wireless Networks and Mobile Communications, Fez, Morocco, 26–29 October 2016. [CrossRef]
13. Arora, K.; Chauhan, R. Improvement in the Performance of Deep Neural Network Model using learning rate. In Proceedings of the Innovations in Power and Advanced Computing Technologies (i-PACT), Vellore, India, 21–22 April 2017. [CrossRef]
14. Tang, T.A.; Mhamdi, L.; McLernon, D.; Zaidi SA, R.; Ghogho, M. Deep Recurrent Neural Network for Intrusion Detection in SDN-based Networks. In Proceedings of the 4th IEEE International Conference on Network Softwarization (NetSoft), Montreal, QC, Canada, 25–29 June 2018. [CrossRef]
15. Yin, C.; Zhu, Y.; Fei, J.; He, X. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. IEEE Access 2017, 5, 21954–21961. [CrossRef]
16. Vinayakumar, R.; Soman, K.P.; Poornachandran, P. Applying convolutional neural network for network intrusion detection. In Proceedings of the International Conference on Advances in Computing, Communications, and Informatics (ICACCI), Udupi, India, 13–16 September 2017. [CrossRef]
17. Zhao, G.; Zhang, C.; Zheng, L. Intrusion Detection Using Deep Belief Network and Probabilistic Neural Network. In Proceedings of the IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC), Guangzhou, China, 21–24 July 2017. [CrossRef]
18. Kim, J.; Kim, J.; Thu HL, T.; Kim, H. Long Short-Term Memory Recurrent Neural Network Classifier for Intrusion Detection. In Proceedings of the International Conference on Platform Technology and Service (PlatCon), Jeju, Korea, 15–17 February 2016. [CrossRef]
19. Staudemeyer, R.C. Applying long short-term memory recurrent neural networks to intrusion detection. S. Afr. Comput. J. 2015, 56, 136–154. [CrossRef]
20. Meng, F.; Fu, Y.; Lou, F.; Chen, Z. An Effective Network Attack Detection Method Based on Kernel PCA and LSTM-RNN. In Proceedings of the International Conference on Computer Systems, Electronics, and Control (ICCSEC), Dalian, China, 25–27 December 2017. [CrossRef]
21. Staudemeyer, R.C.; Omlin, C.W. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. In Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference; Association for Computing Machinery: New York, NY, USA, 2013; pp. 218–224. [CrossRef]
22. Javaid, A.; Niyaz, Q.; Sun, W.; Alam, M. A Deep Learning Approach for Network Intrusion Detection System. In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (BIONETICS), New York, NY, USA, 24 May 2016; pp. 21–26. [CrossRef]
23. Hindy, H.; Brosset, D.; Bayne, E.; Seeam, A.; Tachtatzis, C.; Atkinson, R.; Bellekens, X. A Taxonomy and Survey of Intrusion Detection System Design Techniques, Network Threats and Datasets. arXiv 2018, arXiv:1806.03517.
24. Artificial Neural Network–Wikipedia. Available online: https://en.wikipedia.org/wiki/Artificial_neural_ network (accessed on 29 April 2020).
25. Recurrent Neural Network-Wikipedia. Available online: https://en.wikipedia.org/wiki/Recurrent_neural_ network (accessed on 29 April 2020).
26. Hochreiter, S.; Bengio, Y.; Frasconi, P.; Schmidhuber, J. Gradient Flow in Recurrent Nets: The Difficulty of Learning Long-term Dependencies. In A Field Guide to Dynamical Recurrent Neural Networks; IEEE Press: Piscataway, NJ, USA, 2001; pp. 237–243. [CrossRef]

27. Williams, R.J.; Zipser, D. Gradient based learning algorithms for recurrent networks and their computational complexity. In Backpropagation: Theory, Architectures, and Applications; Lawrence Erlbaum Associates: Hillsdale, NJ, USA, 1995; pp. 433–486.
28. Genetic Algorithms—Introduction. Available online: https://www.tutorialspoint.com/genetic_algorithms/
genetic_algorithms_introduction.htm (accessed on
29 April 2020). 29. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A Detailed Analysis of the KDD CUP 99 Data Set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009;
pp. 1–6.