IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 34, No. 4, August, 2020, pp. 457-463
Journal homepage: http://iieta.org/journals/ria

A Deep Learning Approach to Network Intrusion Detection Using Deep Autoencoder

Srikanthyadav Moraboena[1*], Gayatri Ketepalli[1], Padmaja Ragam[2]

[1] Department of IT, VFSTR (Deemed to be University), Guntur 522213, Andhra Pradesh, India
[2] Department of CS, Andhra Muslim College, Guntur 522003, Andhra Pradesh, India

Corresponding Author Email: kg_itp@vignan.ac.in

https://doi.org/10.18280/ria.340410

Received: 18 June 2020
Accepted: 25 July 2020

Keywords:
deep learning, anomaly detection, autoencoders, NSL- KDD, network security, CICIDS

ABSTRACT

The security of computer networks is critical for network intrusion detection systems (NIDS). However, concerns exist about the suitability and sustainable development of current approaches in light of modern networks. Such concerns are particularly related to increasing levels of human interaction required and decreased detection accuracy. These concerns are also highlighted. This post presents a modern intrusion prevention deep learning methodology. For unattended function instruction, we clarify our proposed Symmetric Deep Autoencoder (SDAE). Also, we are proposing our latest deep research classification model developed with stacked SDAEs. The classification proposed by the Network Security Laboratory-Knowledge Discovery in Databases (NSL-KDD) and Canadian Institute for Cybersecurity -Intrusion Detection System (CICIDS 2017) data sets was implemented in Tensor Flow, a Graphics Procedure Unit (GPU) enabled and evaluated. We implemented and tested our experiment with different batch sizes using Adam optimizer. Promising findings from our model have been achieved so far, which demonstrates improvements over current solutions and the subsequent improvement for use in advanced NIDS.

1. INTRODUCTION

A reliable and efficient network intrusion prevention program is a big problem for network protection. While NIDS technology has made considerable strides, in comparison to anomaly detection strategies, most of the approaches still work with less efficient signature-based technologies. There are several explanations why we fail to move, including the high error rate, accurate training data complexity, reliability of training data, and system behavioral dynamics. The present scenario is going to contribute to an inefficient and unreliable identification by depending on these techniques. The basic features of this problem include the creation of a generally recognized anomaly detection methodology that can solve shortcomings created by the continuing improvements in digital networks.

The use of machine learning, as well as low-level learning methods such as Naive Bayes, Decision Trees, and Support Vector Machinery (SVM) [1], have been a major focus of NIDS research in recent years.

The vast variety of networking and network innovations that have transformed our everyday lives is expected to bring about 50 billion users to the Internet by 2020. For virtually all activities such as internet shopping, finance, business, and email services, these tools are utilized worldwide. Although the benefits of new developments have improved our lives and transformed the environment, the protection of knowledge remains a major concern. Organizations need to provide Internet users, including the customers and staff of organizations, with secure communication channels and detect unlawful activities. Network Intrusion Detection Systems (NIDS), in contrast with other traditional network defense technology such as firewall devices, are currently offering a better approach for the security question. NIDS lets network managers track threats, bugs, and breaches within the network of an organization. The two forms of NIDS are NIDS (SNIDS)-based signature and NIDS (AD-NIDS)-based anomaly. In SNIDS, the system detects attacks by preinstalled rules for NIDS attacks. Data traffic is compared to an updated attack log file to detect network activity violations.

2. BACKGROUND

We should include context details to explain our motives and the ideas behind the model presented in this paper in this section.

2.1 NIDS challenge

With the purposes of protection, forensics, and anomaly detection, network surveillance has been used extensively. A recent development, however, has created several new hurdles to NIDS.

2.1.1 Size

Continues to increase the amount of data collected as well as flowing across networks [2]. The total volume of data is expected to hit 45 ZB by 2020 [3]. As a consequence, new networks have significantly expanded their transmission ability to support the influx of transmission. Most new conduit networks now run at 100 Gigabits per second (Gbps) and higher networking rates. A link of 100Gbps can handle 148,810,534 packets per second to make this context possible [4]. Therefore, a NIDS should be able to complete a packet analysis within 6.72 ns if its operation is at drill speed. It is also difficult to provide NIDS at such a speed and ensure satisfactory levels of accuracy, and efficiency.

2.1.2 Accuracy

Existing techniques cannot be relied on to preserve the aforementioned levels of accuracy. To provide a more holistic and accurate picture, greater granularity, depth, and contextual understanding are necessary. Sadly, this includes the various cost of financing, computing, and time.

2.1.3 Complexity

The amount of different or unique protocols used in digital networks has grown in recent years. The amount of network and/or Web access equipment may be related in half. As a result, differentiation between normal and abnormal traffic and behaviors is becoming increasingly difficult.

2.1.4 Dynamics

The action is complex and challenging to forecast due to the variety and versatility of digital networks. This, in turn, leads to problems with a reliable standard of behavior. There are also questions concerning the lifecycle of learning styles.

2.1.5 Low-frequency assaults

Such forms of threats also disrupted prior methods for anomaly identification, including artificial intelligence. The issue arises from imbalances in the training data collection. In the case of these low-frequency attacks, NIDS has lower detection precision.

2.1.6 Adaptability

New networks have implemented various modern innovations to reduce their dependence on outdated technology and types of management. Dynamic technologies like containerization, virtualization, and software-defined networks are therefore more widely used. NIDS will be prepared to respond to the usage and side effects of these technologies.

2.2 Deep learning

The specialized area of machine learning is Deep Learning, which encourages machine learning similar to artificial intelligence. This allows dynamic interactions and principles to be modeled [5] with several layers. Supervised and unattended learning algorithms help to create higher abstraction rates, which are described using low-level performance characteristics [6].

2.2.1 Autoencoder

Autoencoders, which are used in our proposed solutions, are a common tool commonly used in deep learning science. A neural network extraction algorithm, which learns the best necessary parameters to rebuild its output as close as possible to your input, is an autoencoder. The ability to generalize more powerfully than the Principles Competitor Analysis (PCA) is one of its desirable characteristics.

The recall is used and target values are specified for inputs. In other words, it tries to know the role of identity. The input layer, the output layer, and the secret automotive encoder layer typically have the same input layer. The secret layer is typically smaller than the input. Some researchers use the auto-encoder as a non-linear transformation to investigate fascinating data structures to establish more network constraints and equate findings with pica (linear transformation). This approach is the foundation of the decoder paradigms of the encoder. The input is transformed first and then expanded to replicate the initial data (decoder). The code is given next to the very non-linear input model when you learn the sheet. The following is given. In this model, the dimension of the data entry is the. Because of this, the cornerstone of the deep self-encoder structure is a special layer called an implementation layer [6]. This data layer is used for grouping or mixing purposes in a stacked autoencoder as a compact vector [7]. A low-dimensional version (called coding) of high-dimensional data is generated using the secret layer. The autoencoder must obtain the most important aspects of data flow through this dimension. The data function generated by the autoencoder [8] provides a better summary of the data points in an ideal scenario than the raw information.

2.2.2 Stacked autoencoder

A deep self coding system, unlike a simple autoencoder, has two symmetrical deep creaming networks that have 4 or 5 superficial layers of coding and the second series of 4 or 5 decoding layers. A deep research algorithm has been developed that transforms high-dimensional figures into small-dimensional figures via a deep self-coder using Hinton and Salacukhudinov [9]. Deep knowledge of automatic encoders can be expanded by using a technique called the stacked automatically encoder like the masked layers and several hidden levels of scope. This increased scope decreases cost estimates and the quantity of relevant training information and increases accuracy. The performance is the entry-level of every secret layer that is a step by step higher. Therefore, the raw input typically displays the first sheet of a stacked car encoder key characteristics. The second layer normally learns the second-order characteristics related to patterns on the first-order properties. Later, higher layers know higher functionality. The picture of a stacked self-encoder is an illustrative example. 2. The super-script numbers show here the name of the hidden layer, while the number of subscriptions indicates the dimension of the layer.

3. EXISTING WORK

Zhao et al. [2] submitted an up-to-date computer-safety survey of deep learning technology. They compared conventional computer-learning approaches experimentally with four new methods of deep learning (self-encoders, Boltzmann's small system, convolutional neural network (CNN), and recurrent neural network (RNN)). Their research suggested that revolutionary forms of learning have better consistency than conventional approaches.

Intrusion detection of the network has become the most important part of information security defense network infrastructure. A selection of algorithms is used to identify and distinguish irregularity or assault in NIDS traffic networks, such as a Decision Tree [10], K-nearest neighbor (K-NN) [11], the naive Bayes Network [12], SOM [13], and SVM Network (ANN).

SVM is more efficient than traditional machine learning classification methods [14, 15]. Kim et al.'s job [16] specifically targeted at ongoing and advanced threats. We suggest a robust neural network (DNN) utilizing 100 secret computers, together with the right linear unit activation feature and the ADAM optimizer. They have been built on Tensor Flow's GPU and tested by the KDD. To enhance potential security, the writers reported an overall precision rating of 99 percent. RNN and Large short-term memory (LSTM) models described them.

The research proposed was carried out by analyzing the output of SVM and ANN on the KDD CUP 99 dataset [17]. The findings indicate that SVM experiments are equal to ANN. Consider the Classification and Regression Tree (CART) SVM, naive bayed, logistic regression, decision tree (DT), and KDD CUP 99 data collection of invasive prediction classifications [18]. The tests demonstrated that SVM 's characteristics are distinct.

Unattended methods to research traditional network flow simulation have been brought forward by Cordero etc. [19]. We use the terms RNN, autoencoder, and drop-out. The quality of the procedure you are proposing is not obvious. Via their Fuzziness approach, the authors have established a new perspective focused on intrusion detection semi-monitored research [20]. This approach is based on a random neural weight network and plays a significant part in NIDS diagnosis since computation costs are minimized. The assessment of this model using the NSL-KDD data set was performed, but only binary tasks were examined.

The way network flow data was tracked is also suggested by Tang et al. [20]. There were no specific algorithms in the study, but the NSL-KDD dataset evaluation was 75.75 percent reliable with 6 primary features reported by the publishers. In many unattended training algorithms, SVM, and neural network (NN) was combined to improve the performance of intrusion detectives [21]. To choose the features and SVM or NN for classification, the authors developed, applied, and tested a variety of hybrid models utilizing key parameter analysis (PCA) and GFR gradual reduction. The findings have shown, in terms of preparation and test time, hybrid models are capable of accurately detecting established and unknown threats, and the PCA and GFR classification approaches are expensive to quantify.

The authors have suggested an integral component of a wired or wireless network service for network intrusion detection device (NIDS) for both external and internal assaults [22]. NIDS tracks network-based threats such as malware assaults by Denial of Service (DoS), ransomware spread, and device intrusions.

LSTM is a useful tool for classifying and detecting documented and unknown intrusions [23]. In this review, they suggested a fundamental learning approach to IDS construction. The authors used LSTM RNNs and used NSL-KDD to train the pattern. Despite restricted computational power, the new model has achieved greater accuracy.

4. PROPOSED METHODOLOGY

The proposed architecture indicates the strategies suggested for our experiment. We call to train and testing data sets. The training data collection is marked as friendly or focused for all rows and labels all rows as neutral or an assault form. Then we will apply standardization to this knowledge. After normalization, we educated the data using deep learning methods. We have used standardization for the test data collection and presented these structured data to the IDS model that detects the attack. The proposed architecture is described in Figure 1.

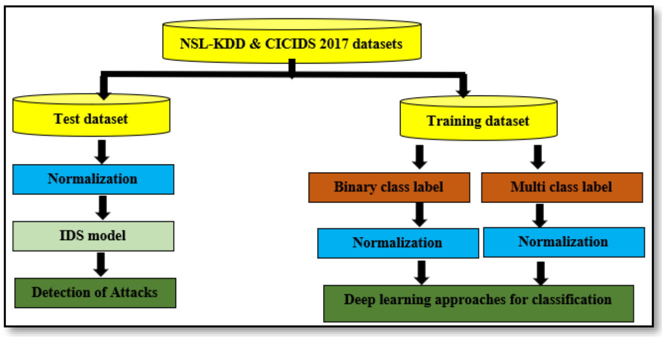

Figure 1. The proposed NIDS architecture

5. RESULTS AND DISCUSSION

Similar to other current deep learning work, Tensor Flow was used to apply our proposed classification algorithm. All our reviews were conducted on Google co laboratory with 13 GB RAM with the GPU-enabled Tensor Flow. We also used the CICIDS and the NSL-KDD datasets to perform our assessments. These two datasets are called landmarks of NIDS science. The usage of such datasets also allows assessing current approaches and studies.

Throughout this portion, we will use the following metrics:

False Positive (FP): Standard data falsely marked as an assault.

True Positive (TP): Attack data properly classified as an assault.

False Negative (FN): Wrongly counted as usual attack results.

True Negative (TN): Normal data correctly categorized as normal. We will use the mentioned measures to assess our proposed solution's performance: The description of performance metrics is shown in Table 1.

Table 1. The performance metrics used

Metric	Description	Formula
ACC	The exactness tests the percentage of the overall amount of false type wrong description.	$\frac{TP+TN}{TP+TN+FP+FN}$
PR	Precision tests the right number of classifications penalized by the sum of false classifications.	$\frac{TP}{TP+FP}$
RE	The recall measures the right amount of penalized classifications The number of submissions missing.	$\frac{TP}{TP+FN}$
FA	The false alarm wrongly tests the ratio of positive occurrences categorized as malevolent.	$\frac{FP}{FP+TN}$
F S	The F-score tests the harmonic mean of precision and recall which serves as an efficiency measurement derived.	$2*\frac{Precision*Recall}{Precision+Recall}$

5.1 Datasets

The NSL-KDD and CICDS2017 contrasts are included in this article. For IDS research both natural and abnormal partnerships were widely used.

NSL-KDD: The more recent NSL-KDD dataset [24] developed by Patil and Srikanth Yadav [25] to solve KDD 99 data collection problems.

The authors [26] have done a taxonomy survey of the deep architectures and algorithms accessible in these works and grouped such algorithms into three groups: hierarchical, composite, and generative. Afterward, a wide range of intrusion detection fields investigates selected deep learning applications.

Some of the current NIDS studies utilize this dataset as well, and we hope that researchers will compare various strategies. Table 2 displays the NSL-KDD data set's instruction and evaluation documentation collection.

Table 2. Various forms of NSL-KDD data collection attacks

Attack type	Flow count	Training	Test
Normal	12697529	67354	9734
DoS	659050	45935	7469
Probe	281156	11678	2405
R2L	114596	991	2748
U2R	201889	49	213
Total	13954220	126007	22569

In the 2017 CICIDS datasets [27], the evaluation CICIDS dataset collects 80 Network Flow features from the network traffic created. It includes SSH, DoS, Heart Blood, Hack, Botnet, DDoS, and Brute ForceFTP. 80 network flow properties are extracted from network traffic generated from CIC traffic flow. The CIC-IDS2017 dataset also comprises 25 specific implementations, including FTP and HTTPS. Based on other standards, Table 3 displays the collection of CICIDS data sets for testing and training.

Table 3. Various forms of CIC-IDS2017 data collection attacks

Category of Attack	Type of attack	Flow count	Training	Test
Brute-force Web attack	SSH	241	175	42
	FTP	601	477	112
	XSS	187599	7525	1856
	Web	193360	15532	3850
	SQL Injection	86	65	13
DoS attack	Hulk	466654	18658	4658
	SlowHTTPTest	139880	55947	13996
	Slow Loris	10994	4377	1085
	Goldeneye	41522	16503	4162
DDoS attack	HOIC	686112	27445	6860
	LOIC-UDP	1736	1356	339
	LOIC-HTTP	576291	23146	5755
Botnet	Bot	286191	11385	2962
Infiltration	Infiltration	161933	6477	1633
Benign		12697529	50995	12678
Total		15450729	240063	60001

In general, the analysis reveals that by studying feature representations from broad quantities of unlabeled training samples the proposed model will achieve high efficiency. The session-based training samples are constructed from header sections and network packet loading information. We observed that the deep learning approach that was introduced obtained very strong results for various classification tasks. These results provide insights into the characteristics of raw traffic. Such function representations are successful in detecting specific malicious network traffics and creating low false alarms. The following recommendations for reconstruction error and true class metrics for various batch sizes 32, 64, 128, 256, 512, and 1024, as shown in Tables 4, 5, and 6 have been considered in 74257 protocol-type CICIDS samples.

Table 4. Reconstruction error and true class for batch size 32 and 64

	Batch size: 32		Batch size: 64	
	reconstruction error	true class	reconstruction error	true class
count	74257	74257	74257	74257
mean	0.756783	15.768951	0.612605	15.768951
std	16.011881	4.52167	15.536912	4.52167
min	0.030319	0	0.014329	0
25%	0.113532	14	0.088711	14
50%	0.194001	16	0.122684	16
75%	0.412455	16	0.294209	16
max	2026.624732	39	1992.387512	39

Table 5. Reconstruction error and true class for batch size 128 and 256

	Batch size: 128		Batch size: 256	
	reconstruction error	true class	reconstructio n error	True class
count	74257	74257	74257	74257
mean	0.541016	15.768951	0.505147	15.768951
std	11.985496	4.52167	11.084156	4.52167
min	0.018649	0	0.019048	0
25%	0.085526	14	0.084922	14
50%	0.115952	16	0.114374	16
75%	0.273082	16	0.273093	16
max	1788.619462	39	1556.290473	39

Table 6. Reconstruction error and true class for batch size 512 and 1024

	Batch size: 512		Batch size: 1024	
	reconstruction error	true class	reconstruction error	true class
count	74257	74257	74257	74257
mean	0.490655	15.768951	0.485803	15.768951
std	10.675083	4.52167	10.519899	4.52167
min	0.018295	0	0.018811	0
25%	0.085305	14	0.084722	14
50%	0.114188	16	0.113876	16
75%	0.273149	16	0.272969	16
max	1629.798801	39	1637.784804	39

Figure 2 below shows the F1-Scores for different batch sizes; Figure 3 below shows the ROC curves for different batch sizes; Figure 4 below represents a model loss for different batch sizes.

Table 7 describes the comparison results of training dataset upon calculation of model loss for all epoch, F1-score, and ROC curve of TPR and FPR. The F1-score results are depicted in Figure 2, TPR and FPR results are portrayed in Figure 3, and the model loss results of all epochs are depicted in Figure 4.

Table 8 demonstrates accuracy and failure for different batch sizes. After reducing features, we have ultimately considered 14 factors for our experiment. In our experimentation with 3 encoders and 3 decoders, in each stage, the preceding encoder or decoder is provided as an input iteratively and in any iteration, we take into account the encoding dimension size by 2. For batch size 32 we have a loss of 0.6874 and an accuracy of 0.4684. For 64 batch size, we have 0.9144 accuracy with a loss of 0.5528. We have 0.9144 accuracy with a loss of 0.5177 with 128 batch capacity. Given 256 batch size, we have an accuracy of 0.9157 with a loss of 0.4752. We have 0.9164 accuracy with 512 batch size and 0.4651 loss. Eventually, we received 0.9176 accuracy with 0.4627 failure with batch size 1024.

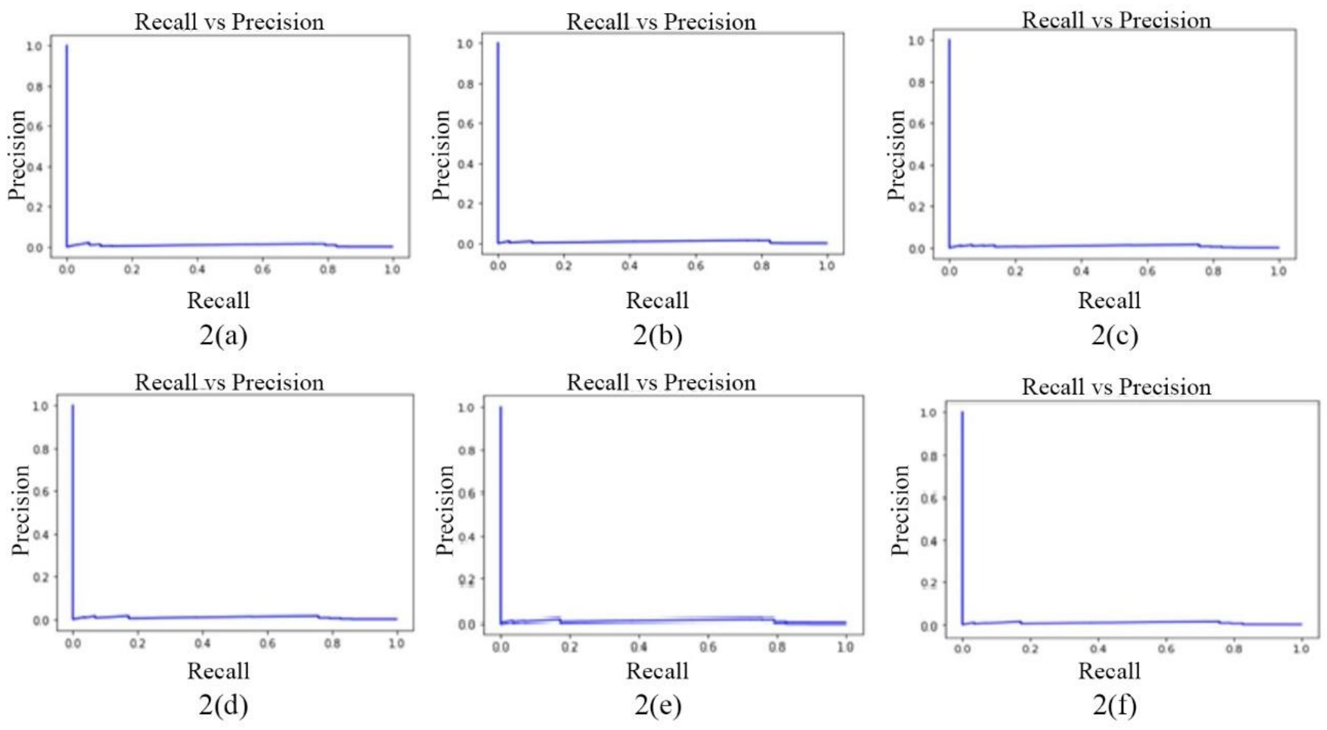

Figure 2. 2(a) to 2(f) F1-Scores for batch sizes 32, 64, 128, 256, 512 and 1024 respectively

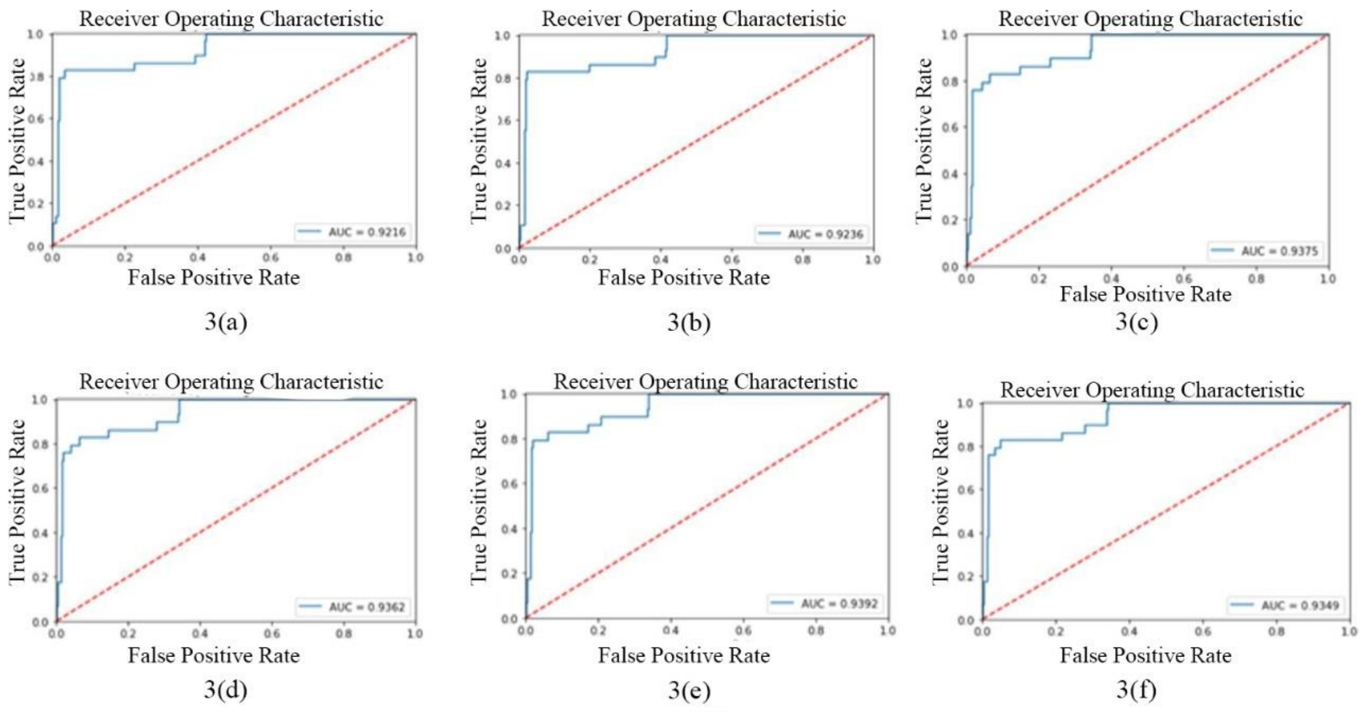

Figure 3. 3(a) to 3(f) ROC curves for batch sizes 32, 64, 128, 256, 512 and 1024 respectively

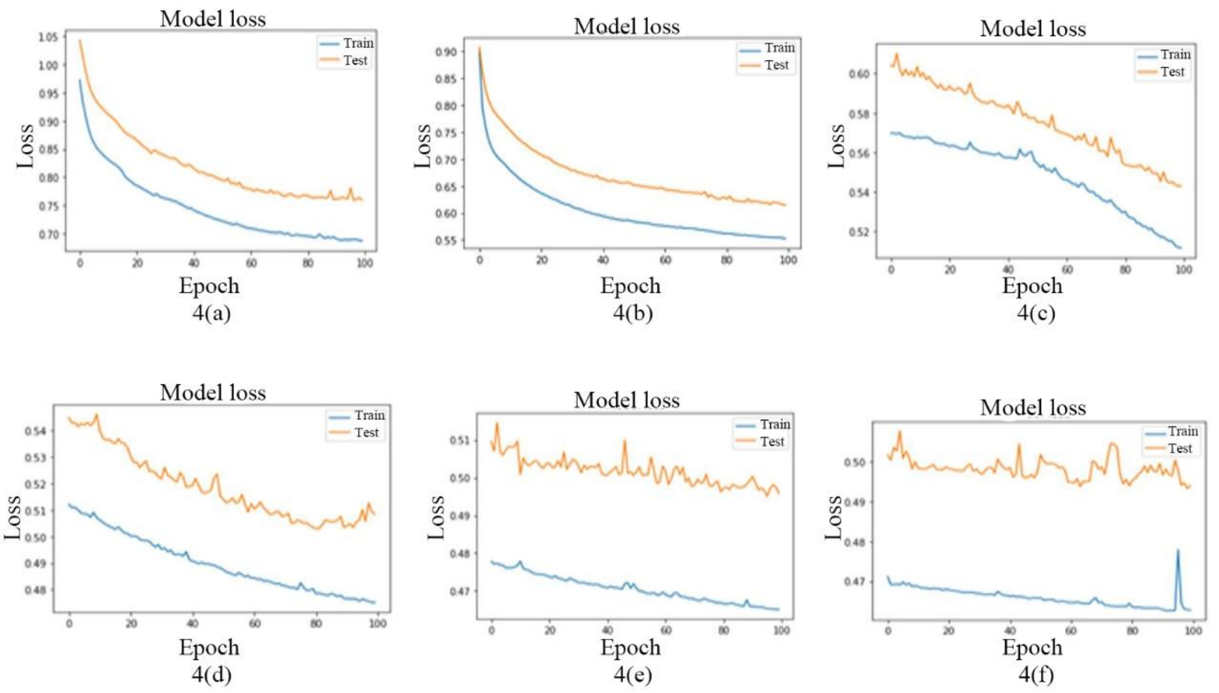

Figure 4. 4(a) to 4(f) ROC curves for batch sizes 32, 64, 128, 256, 512 and 1024 respectively

Table 7. Comparison results of model loss, F1-score, TPR, and FPR

Batch size	**Model loss & Accuracy**		**F1-scores**	
	Loss	Accuracy	Precision	Recall
32	0.6874	0.4684	0.9556	0.9687
64	0.5528	0.9144	0.9866	0.9629
128	0.5117	0.9144	0.9699	0.9594
256	0.4752	0.9157	0.9699	0.9594
512	0.4651	0.9164	0.9556	0.9687
1024	0.4627	0.9176	0.9627	0.9640

Table 8. Loss and accuracy for different batch sizes with Adam optimizer

Batch size	**No.of epochs**	**Loss**	**Val_ Loss**	**Accuracy**	**Val_ Accuracy**
32	100	0.6874	0.7596	0.4684	0.4607
64	100	0.5528	0.6145	0.9144	0.8256
128	100	0.5117	0.5432	0.9144	0.8897
256	100	0.4752	0.5085	0.9157	0.9124
512	100	0.4651	0.4958	0.9164	0.9188
1024	100	0.4627	0.4939	0.9176	0.9183

6. CONCLUSION

The experimental results of the method proposed indicate that our model shows improved classification accuracy and faster training set and testing time. The methods have achieved a higher precision with batch sizes 512 and 1024, especially in binary classification with 0.9176 accuracies and 0.9188 value accuracy of the CICIDS dataset. The further development will be the subject of our future expansion strategy by using a hybrid feature learning model for a great indication for reduction of dimensionality and classification mechanisms. Also, the training and test times of the model should be further minimized with parallel structures or GPU acceleration of the device.

REFERENCES

[1] Dong, B., Wang, X. (2016). Comparison deep learning method to traditional methods using for network intrusion detection. 8th IEEE International Conference on Communication Software and Networks (ICCSN), Beijing, pp. 581-585. http://dx.doi.org/10.1109/ICCSN.2016.7586590

[2] Zhao, R., Yan, R.Q., Chen, Z.H., Mao, K.Z., Wang, P., Gao, R.X. (2019). Deep learning and its applications to machine health monitoring. Mechanical Systems and Signal Processing, 115: 213-237. https://doi.org/10.1016/j.ymssp.2018.05.050

[3] IDC. (2014). Executive summary: Data growth, business opportunities, and the IT imperatives. Framingham, MA, USA, Tech. https://www.emc.com/leadership/digital universe/2014iview/executive-summary.htm, accessed on 17 June 2020.

[4] Juniper, N. (2015). Juniper Networks—how many packets per second per port are needed to achieve Wire-Speed? https://kb.juniper.net/InfoCenter/index?page=content&id=KB14737, accessed on 17 June 2020.

[5] Heaton, N. (2018). Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep learning. Program Evolvable Mach, 19: 305-307. http://dx.doi.org/10.1007/s10710-017-9314-z

[6] Li., D., Yu, D. (2014). Deep learning: Methods and applications. Deep Learning: Methods and Applications, 206. http://dx.doi.org/10.1561/9781601988157

[7] Pascal, V., Hugo, L., Isabelle, L., Yoshua, B., Pierre-Antoine, M. (2010). Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. The Journal of Machine Learning Research, 11: 3371-3408.

[8] Javaid, A., Niyaz, Q., Sun, W., Alam, M. (2016). A deep learning approach for network intrusion detection system. Proc. 9th EAI Int. Conf. Bio-Inspired Inf. Commun.

Technol., pp. 21-26. http://dx.doi.org/10.4108/eai.3-12-2015.2262516

[9] Wang, Y.S., Yao, H.X., Zhao, S.C. (2015). Auto-Encoder based dimensionality reduction. neurocomputing. ACM Journals, 184. https://doi.org/10.1016/j.neucom.2015.08.104

[10] Kim, G., Lee, S, Kim, S. (2014). A novel hybrid intrusion detection method integrating anomaly detection with misuse detection. Expert Systems with Applications, 41:1690-1700. http://dx.doi.org/10.1016/j.eswa.2013.08.066

[11] Koc, L., Mazzuchi, T, Sarkani, S. (2012). A network intrusion detection system based on a hidden naïve bayes multiclass classifier. Expert Systems with Applications, 39: 13492-13500. http://dx.doi.org/10.1016/j.eswa.2012.07.009

[12] De la Hoz, E., De la Hoz, E., Ortiz, A., Ortega, J., Martínez-Á, A. (2014). Feature selection by multi-objective optimization: Application to network anomaly detection by hierarchical self-organizing maps. Knowledge-Based Systems, 71: 322-338. http://dx.doi.org/10.1016/j.knosys.2014.08.013

[13] Rinku, S., Manojit, C., Nilanjan, S. (2015). An efficient approach to develop detection system based on multi-layer Backpropagation Neural Network Algorithm: IDS using BPNN Algorithm. ACM SIGMIS Conference on Computers and People. Research. Association for Computing Machinery, New York, NY, USA, pp. 105–108. https://doi.org/10.1145/2751957.2751979

[14] Guang, K., Yi, P., Chen, Z., Yong, S. (2009). Multiple criteria mathematical programming for multi-class classification and application in network intrusion detection. Information Sciences, 179: 371-381. https://doi.org/10.1016/j.ins.2008.10.025

[15] Mehmood, T., Rais, H. (2015). SVM for network anomaly detection using ACO feature subset. 2015 International Symposium on Mathematical Sciences and Computing Research (iSMSC), Ipon, pp. 121-126, https://doi: 10.1109/ISMSC.2015.7594039

[16] Kim, J., Shin, N., Jo, S., Kim, S. (2017). Method of intrusion detection using deep neural networks. 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju, pp. 313-316 https://doi.org/10.1109/BIGCOMP.2017.7881684

[17] Long, C., Fei, Y., Long, J., Zheng, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access, 5: 21954-21961. https://doi.org/10.1109/ACCESS.2017.2762418

[18] Garcia, C., Carlos, H., Sascha, M., Max, F., Mathias. (2016). Analyzing flow-based anomaly intrusion detection using Replicator Neural Networks. 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, pp. 317-324. https://doi.org/10.1109/PST.2016.7906980

[19] Ashfaq, R., Wang, X., Huang, J., Abbas, H., He, Y. (2017). Fuzziness based semi-supervised learning approach for Intrusion Detection System. Information Sciences, 378: 484-497. https://doi.org/10.1016/j.ins.2016.04.019

[20] Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M. (2016). Deep learning approach for network intrusion detection in software defined networking. 2016 International Conference on Wireless Networks and Mobile Communications (WINCOM), Fez, pp. 258-263. https://doi.org/10.1109/WINCOM.2016.7777224

[21] Perez, D., Astor, M., Perez, A, David, S., Eugenio. (2017). Intrusion detection in computer networks using hybrid machine learning techniques. 2017 XLIII Latin American Computer Conference (CLEI), Cordoba, pp. 1-10. https://doi.org/10.1109/CLEI.2017.8226392

[22] Srikanth Yadav, M., M.C.P/ Saheb. (2020). Comparative assessment of deep learning methods for network intrusion detection. Journal of Critical Reviews, 7(18): 314-320.

[23] Srikanth Yadav, M., Sushma, K., Gayatri.K. (2020). Enhanced network intrusion detection using LSTM RNN. International Journal of Advanced Science and Technology, 29(05): 7210-7220.

[24] Nslkdd. https://www.unb.ca/cic/datasets/nsl.html, accessed on May 30, 2019.

[25] Patil, A., Srikanth Yadav, M. (2018). Performance analysis of misuse attack data using data mining classifiers. International Journal of Engineering & Technology, 7 (4.36): 261-263. ISSN 2227-524X. http://dx.doi.org/10.14419/ijet.v7i4.36.23782

[26] Cicds. Dataset. https://www.unb.ca/cic/datasets/ids-2017.html, accessed on May 30, 2019.

[27] Srikanth Yadav, M., Kalpana, R. (2019). Data preprocessing for intrusion detection system using encoding and normalization approaches. 2019 11th International Conference on Advanced Computing (Icoac), Chennai, India, pp. 265-269. https://doi.org/10.1109/Icoac48765.2019.246851