Using Deep Learning Techniques for Network Intrusion Detection

Sara Al-Emadi, Aisha Al-Mohannadi, Felwa Al-Senaid
Department of Computer Science and Engineering
Qatar University
Doha, Qatar
Email: saraalemadi@ieee.org, [aa1401818, fa1003489]@qu.edu.qa

***Abstract*—In recent years, there has been a significant increase in network intrusion attacks which raises a great concern from the privacy and security aspects. Due to the advancement of the technology, cyber-security attacks are becoming very complex such that the current detection systems are not sufficient enough to address this issue. Therefore, an implementation of an intelligent and effective network intrusion detection system would be crucial to solve this problem. In this paper, we use deep learning techniques, namely, Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) to design an intelligent detection system which is able to detect different network intrusions. Additionally, we evaluate the performance of the proposed solution using different evaluation matrices and we present a comparison between the results of our proposed solution to find the best model for the network intrusion detection system.**

***Index Terms*—Network Intrusion Detection, Deep Learning, Neural Network, Recurrent Neural Network, RNN, Convolutional Neural Network, CNN, Network Security**

I. Introduction

As technology develops, new and complex cyber-attacks are being deployed to break through systems in order to exploit vulnerabilities and other malicious activities. Network infrastructure is one of the main systems which experiences a dense quantity of different types of cyber-attacks such as Denial of Service (DoS) or distributed denial-of-service attacks (DDoS), TCP SYN Flood attack, Ping of death attack, Teardrop attack, Scan attacks, etc. Hence, it has been observed that a great amount of effort was put in finding and implementing different methods and techniques to block these attacks and ensure that the network is safe and secure while maintaining high availability to the legit users of the network.

A well-known method of securing the network is through implementing an intrusion detection system (IDS). IDS was originally implemented in 1980 by the authors in [1]. The main aim of their work was to introduce a mechanism which differentiate between benign activities from malicious ones. Further research was carried out to optimizing this methodology to aid monitoring the network traffic in case of attacks, this system is now known as Network Intrusion Detection System (NIDS) [2]. In NIDS, the detection system is inspecting the incoming and outgoing network traffic from all hosts in real time and based on certain criteria, it can detect and identify the attack, then, take the suitable security measures to stop or block it, which significantly reduces the risk of damage to the network. However, due to the rapid increase in the complexity of the cyber-security attacks, the current methods used in NIDS are failing to sufficiently address this issue. Therefore, in this paper, we aim to design an intelligent detection system using different Deep Learning algorithms, namely, Recurrent Neural Network (RNN) and the Convolutional Neural Network (CNN) which is capable of autonomously detecting, identifying and differentiating between different network intrusion attacks.

A recent systematic review about the use of Artificial Neural Networks(ANN) for NIDS identified that there is a need for further studies in the use of deep neural networks and its variants[3]. In the recent years, employing of Deep Neural Networks in IDS has gained popularity due to its robustness and usefulness. The NIDS today are facing various challenges such as high-level feature extraction. Studies show the usefulness of Machine learning or Deep Learning algorithms in attacks classifications. A study by Vinyakumar, Soman, and Poornachandran (2017) highlights that the Deep Learning Networks are known for its "hierarchical feature representations, learning long-term dependencies of temporal patterns in large scale sequence data" [4].

Deep Learning Network architectures such as CNN and RNN can overcome the complexities of existing classifiers. Furthermore, Deep Learning Networks are known for their high accuracy and improved performance, thus can be a critical component in Network Intrusion Systems. Therefore, in this study, we have chosen CNN based on its nature and effectiveness in classifying objects which could be utilised in the detecting normal and malicious traffics process. On the other hand, RNN was selected due to its features of remembering previous occurrences which could contribute to a crucial performance in classifying consequence multiple types of attacks. The principal objective of this research is to study the effectiveness of implementing CNN and RNN in NIDS. The study will conduct an experimental simulation to analyze the proposed system's performance in intrusion detection.

The contribution of this paper can be summarized as follows:

- Providing a study on CNN and RNN in network intrusion detection systems.
- Evaluating the performance of the deep learning algorithms in NIDS by analysing the accuracy, precision, recall, and F1 score for each.

- Providing a comparison between the results found through the proposed solution with the literature with different dataset.

The rest of the paper is organized as follows; Section II discuss the state-of-the-art solutions using NIDS, Section III explains the project design and illustrates the architecture of the proposed solution, whereas the implementation is conducted in Section IV-B, and experimental results discussion in section V.

II. Related Work

In previous studies, there were different approaches used for intrusion detection systems. Authors in [5], tackled an improved intrusion detection scheme which was a three layered RNN that is based on different features. The inputs for the model were features that are classified based on basic features, content features, time-based traffic features, and host-based traffic features and the output of the model is either classifying the normal class which indicates that there are no attacks or classifying attacks such as DoS, R2L, U2R and probing. Furthermore, they explained how they used KDD dataset to train and test the model and the connections between the hidden layers that are partially connected which speeds the process of classification. The technique used for this paper is misuse-based intrusion detection which compares user's activities with the known behaviors of attackers.

Another approach was introduced in [6] to detect network intrusion using Gated Recurrent United Recurrent Neural Network (GRU-RNN). In their work, the authors have attempted to provide a novel solution for Network Intrusion Detection (NID) in Software Defined Networks (SDN) using very low number of features. The authors claimed that the advantage of using GRU-RNN in comparison to the traditional RNN or LSTM is that it avoids vanishing and exploding gradient problems. Furthermore, the authors have implemented their system as an application on the SDN controller using NSL-KDD dataset for training and testing their model. The GRU-RNN proposed have outperformed other machine learning algorithms such as the VanillaRNN, SVM, DNN with detection rate of 89% for legitimate events and 90% for the anomaly events. Additionally, the author claimed that the overall accuracy of their proposed solution is 89% which is the highest in comparison to the other state-of-the-art algorithms discussed.

Moreover, authors in paper [7] examined and proposed a method for the intrusion detection using Neural Networks which utilizes the concepts of Deep Brief Network (DBN), probabilistic neural network (PNN), and particle swarm optimization (PSO) algorithm as it evolves four steps. In the initial step, they used the concept of deep learning as the raw data translates into low dimensional data thus reflecting the characteristics of the data. This is achieved through a multi-level or deep neural network. Also, those DBN are formed through multi stacking of Restricted Boltzmann Machine (RBM)s which ensure the normalization of the data. In the next step, the authors optimized the data using the PSO algorithm which optimizes the hidden layer nodes of data that facilitate the learning performance of network. Finally, training and testing will be performed by feeding data into a PNN as it uses a local approximation network that uses Gaussian function and the network activation function, thus produces an optimized data.

In addition, authors in [8] used hybrid detection method that is based on a combination of RNN with restricted Boltzmann machines (RBM) as they begin with taking multi-layer RBM model with byte level raw data inputs and then obtain network packets feature vectors. Afterwards, a sequence of packets is modelled by the RNN model which creates the micro flow features. The proposed model takes raw data input without feature engineering. Finally, a softmax classification is applied in order to distinguish if a micro-flow is malicious or not. Also, in order to identify how many layers of RBM features are needed, they performed a series of experiments on the effects of the 1-5-layer RBM model, including the convergence rate and various evaluation indicators. And they have pointed out that as the number of layers in the RBM model is improved, the convergence rate decreases, and more epochs are needed to obtain better results. That is why they recommended to use 3 layered RBM model.

Using CNN algorithm, a study was made by authors in [9] that discusses and investigates the network intrusion detection using the Convolutional Neural Networks (CNN). For the purpose of study, behavior-based classifier learning model with CNN has been developed. The CNN along with TensorFlow and SoftMax function is utilized to extract the behavioral features and recognize the class of threats using statistical data. During the extraction phase, a benchmark of data is created sources such as, an archive dataset of behavior features from KDD Cup99 and Suspicious network flow order by NHSNC. During the model learning phase, the current study uses a revised model of LeNet-5 model along with gradient-descent optimization algorithm. These algorithms facilitate to fine-tune the model parameters using both error derivatives of back propagation and the learning rate for all layers. In a nutshell, these classification helps to draw the learning errors of multiple layer neural nets and to minimize the weights of neural network which eventually have an impact on the learning process of CNN.

III. Deep learning models

The proposed intrusion detection system uses the concept of two major Deep Neural Networks architectures which are the CNN and RNN. The CNN and RNN security enabled system architecture addresses the identification and classifications of attacks. In this paper, we use two types of RNNs algorithms which is the Long short-term memory (LSTM) and the Gated Recurrent Units (GRU). In LSTM, it has the ability to process single data as well as complete sequence of data. Also, it has the advantage of negligence to gap length. Moreover, Gated Recurrent Units (GRU) is similar to LSTM layers however they lack output gates and they have less parameters than LSTM does as GRU are a kind of LSTM. In case of CNN which is known as a feed forward Neural Network, it processes only the current input. Clearly, every layer in the architecture has its own functionality that determines the hidden layers and implements feature extraction.

In this project, we use CNN architecture which has the advantage of feature selection and extraction without human interaction. The CNN architecture implemented with one convolutional layer followed by a max pooling layer that reduces the number of parameters when the dataset is initially fed to the algorithm, then a dense layer is applied in order to enable connections between neurons with other layers and finally a dropout layer is used in order to avoid any overfitting. In case of RNN architecture, two layers of LSTM is used which has an ability of remembering their inputs over a long period of time. Also, after each LSTM layer a dropout layer is applied and finally the dense layer followed by the activation layer. The same architecture is used for the GRU approach only instead of using LSTM layers, it was changed to GRU layers. Figure 1 shows the processing of the data when picked up from the monitor and then being analyzed to distinguish if it was a normal packet or not as this figure was an inspiration by the architecture in [10].

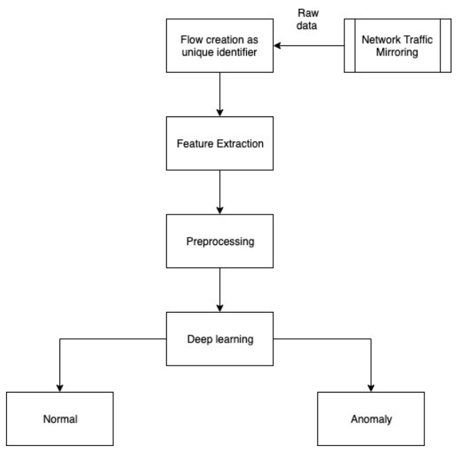

Fig. 1. Deep learning models architecture

IV. Methodology

A. Dataset

Implementing IDS with Deep Learning require a quality amount of dataset to train and test the algorithm which resembles the real time environment. NSL-KDD, a standard data set available from Canadian Institute of Cybersecurity[1], is used in the current paper for training and testing. This dataset ensure improved performance due to the absence of redundancy in both train and test sets. The dataset contains only a reasonable number of traffic records, which facilitate to run the experiment on the complete set rather than a portion of the set. Out of all dataset, about 85% are specifically allocated for training purpose while the remaining 15% for the testing and validation purpose. In the NSL-KDD, each records encompasses 41 attributes of different features and a label to identify the traffic as malicious or normal. Table I demonstrate the 41 attributes of records in NSL-KDD dataset. The attributes are further categorized based on its core features into four categories: such as basic, content, time, host. In the following section, this paper will provide a brief overview of each category with some examples.

TABLE I
LIST OF NSL-KDD DATASET RECORDS ATTRIBUTES [11]

Category	No.	Attribute Name
Basic Features	1	duration
	2	protocol_type
	3	service
	4	flag
	5	src_bytes
	6	dst_bytes
	7	land
	8	wrong_Fragment
	9	urgent
Content Related Features	10	hot
	11	num_Failed_Logins
	12	logged_In
	13	num_compromised
	14	root_Shell
	15	su_attempted
	16	num_root
	17	num_file_creations
	18	num_shells
	19	num_access_files
	20	num_outbound_commands
	21	is_host_login
	22	is_guest_login
Time Related Features	23	count
	24	srv_count
	25	serror_rate
	26	srv_error_rate
	27	rerror_rate
	28	srv_rerror_rate
	29	same_srv_rate
	30	diff_srv_rate
	31	srv_diff_host_rate
Host Based Traffic Features	32	dst_host_count
	33	dst_host_srv_count
	34	dst_host_same_srv_rate
	35	dst_host_diff_srv_rate
	36	dst_host_same_src_port_rate
	37	dst_host_srv_diff_host_rate
	38	dst_host_serror_rate
	39	dst_host_srv_serror_rate
	40	dst_host_rerror_rate
	41	dst_host_srv_rerror_rate

First category, known as basic, represents the network connection attributes. This category consists of nine attributes which represent the fundamental connection features including protocol type, connection status, and number of bytes transferred between source and destination. The protocol type label in this group refers to the protocols which may have used at the application layer like TCP/IP, UDP or ICMP. Moreover, this category consist of an attribute, named as Wrong-fragment, represent the total number of wrong fragments in the connection. Likewise Urgent attribute refers to the number of urgent packets which will have the urgent

[1]https://www.unb.ca/cic/datasets/nsl.html

bit activated for identification. Similarly, the second category of attribute refers to the content related features of the data records such as Number of failed logins (Num_failed_logins), number of compromised situations(Num_compromised), number of file creation(Num_file_creations), number of shell prompts (Num_shells), and guest login identification (Is_guest_login)[12]. The attribute representing number of failed logins can essentially contribute to understand the malicious possibilities associated to the record. In addition, most of the content related features helps the system to learn the abnormal characteristics of traffic using the content of data records. The third category of features illustrates the time associated traffic features of data records. The corresponding category consisting of count of connection to same ends like destination host and service port number. Additionally, this category also include attribute which hold the percentage of connections that have activated special flags for various error rate. The next category of attribute corresponds to the host based traffic features such as number of connection with same destination host, number of connection with same port number, percentage of connection to same service and percentage of connection to different services. This category include almost ten attributes with various functionalities. Every record in dataset end with a label, which often considering as the 42nd attribute, stating whether the record is a normal or malicious activity. The label in turn contribute to the system to learn and validate the traffic efficiently. In general, NSL-KDD dataset attribute values are represented in nominal, binary, and numeric depending on the type of data it holds. For the purpose of study, the datasets are preprocessed to meet the requirements of a machine learning system. In particular, every value in a single entry of the dataset was converted to numeric format to facilitate the needs of the proposed system.

B. Implementation

In the implementation phase, we have used the open source NSL-KDD dataset [13] which is an enhanced version of the well-known KDD'99 dataset [14]. This dataset consists of different network intrusion attacks such as U2R and R2L attacks. Additionally, we have utilized and enhanced on the code provided in [15] for both CNN and RNN to suit the needs of the experiments carried out through the research.

As part of implementing this solution, we ran the experiments multiple times with different parameters in order to find the optimal hyperparameters which will ensure that we leverage the highest performance of the models. Doing so, we changed the output filter in the LSTM layer by increasing it to 256 as this results in increasing the accuracy. Furthermore, the models were trained with 1000 epochs. The dataset distribution was divided into multiple portions; 85% was used for the training process and 15% was used for validation and testing.

V. Performance Evaluation

The detection of network intrusion attack is considered a binary classification problem in which the system will be trained and tested to differentiate between a malicious packet and a benign one. In this section, we will further explain and demonstrate the use of deep learning technique in implementing the NIDS and the effectiveness of its performance.

A. Experimental Setup

In order to implement the solution proposed in section IV-B, a suitable environment setup, shown in Table II, was used to run the deep learning algorithms in training, validation and testing phases.

TABLE II
Environment setup

Operating System	**Ubuntu 18.04-Linux**	**MacOS 10.14.4**
GPU	Nvidia Titan V	-
CPU	2.10 GHz Intel Xeon	2.9 GHz Intel Core i5
Number of CPU	36	-
Framework/APIs	Python 3.6	Keras APIs

The first workstation(Ubuntu 18.04 Linux) was used to run the experiments of CNN algorithm, whereas the second workstation (MacOS 10.14.4) was used to run both RNN-LSTM and RNN-GRU experiments. It is important to note that both workstations had similar performance in terms of speed and time consumption for all the experiments carried out in this paper.

The evaluation of the deep learning model performance computed in the testing phase was based on four main evaluation metrics, which are:

- **Accuracy**: is used to find ratio of correct predictions to total number of predictions.
- **Recall**: provides an overview on the sensitivity of the model, that is, the ratio of the positive data which was correctly identified as positive to the overall positive data.
- **Precision**: resembles the ratio of the correctly predicted data to the overall positively predicted data, hence, a model with high precision is able to identify majority of the predicted data correctly.
- **F1 score**: shows the overall performance of the model in correlation to both precision and recall. The advantage of using F1 score over accuracy to measure the overall performance of a model is that it takes into consideration the distribution of data and the scenario of uneven classes where the false positive and false negatives are at play. Moreover, it is important to use it in this application as it will give us a better understanding of the system.

These metrics were selected based on the fact that CNN, RNN-LSTM and RNN-GRU are deep learning algorithms with different characteristics and we expect that each of these algorithms will excel at one of multiple of the evaluation matrices. For example, due to the nature of RNN algorithms of remembering occurrence of previous events, we expect that RNN-LSTM and RNN-GRU to outperform CNN in the recall value.

B. *Experimental Results*

After the implementation of the deep learning architectures using the open-source code in [15] which we extensively modified to suit the needs of our study, we ran the experiments for CNN, RNN-LSTM and RNN-GRU multiple times to observe and analyze their performance. These experiments were undertaken using the training procedure described in algorithm 1. In the training phase, the validation dataset is used to measure the accuracy of the algorithm at every epoch. Therefore, the model is stored only when the validation accuracy of the current epoch is higher than the best accuracy measured so far, once this case occurs, a copy of the model at this epoch is stored which can be used later on in the testing phase.

Algorithm 1 Deep Learning Model Training Algorithm

```
0:  procedure TRAINING PROCEDURE
0:     TrainingDataset ← Preprocessed KDDTrain
0:     ValidationDataset ← Preprocessed KDDTest
0:     $i \leftarrow 0$
0:     $epoch \leftarrow 1000$
1:  while $i \neq epoch$ do
2:     Training Phase for 1 epoch with Cross Validation
3:     Evaluate on Unseen data
4:     if $ValidationAcc$ is 0 OR $i$ is 0 then
5:        $BestValidationAcc \leftarrow ValidationAcc$
6:     end if
7:     if $ValidationAcc > BestValidationAcc$ then
8:        $BestValidationAcc \leftarrow ValidationAcc$
9:        Saves Model_$i$.hf5
10:    end if
11:    $i \leftarrow i + 1$
12: end while
```

Several hyperparameters of training the Deep Learning algorithms were defined based on heuristic experiments. As mentioned in section IV-B, the training phase ran for 1000 epochs for each experiment. However, it has been observed that each algorithm converges by reaching its optimal performance before 1000 epochs as indicated in Table III. This observation indicates that in NIDS where there is limited computational power, these algorithm can achieve great output while being trained in short time with number of epochs less than 1000, hence, they are considered very efficient.

TABLE III
OPTIMUM NUMBER OF EPOCHS

Algorithm	*Epochs*
CNN	598
LSTM	987
GRU	879

The experiments carried out throughout this work are summarized in Table IV. From the result yielded from these experiments, one can deduce that CNN have substantially outperformed LSTM and GRU in terms of F1 score, precision and accuracy with the lowest number of epochs required to reach its optimal performance. However, another promising finding was the results of RNN-LSTM and RNN-GRU in recall matrix, as both of these algorithms have significantly outperformed CNN. This is due to their architecture nature, since both models are RNN models and have the characteristic of remembering previous occurrence. Also, it can be concluded from the results found that RNN-GRU algorithm is had the weakest performance among the other two. Hence, using this Deep Learning techniques without tuning the hyper-parameters would not be beneficial in the NID process.

TABLE IV
NID RESULTS

Metric	*CNN*	*RNN-LSTM*	*RNN-GRU*
F1 Score	98.48%	89.54%	65.53%
Precision	100%	81.24%	48.77%
Recall	97.01%	99.74%	99.88%
Accuracy	97.01%	81.60%	50.25%

We have verified throughout this study that Deep Learning techniques are useful and can play a key role in NIDS. Although we did not conclude the exact results which were previously reported in the literature [4] due to a number of factors; such as the difference in the dataset used given that we have used NSL-KDD and the authors of [4] opted for KDDCup'99 and the difference in hyperparameters of the Deep Learning models, yet, both studies agree on effectiveness of using Deep Learning Techniques for NIDS. Furthermore, our results suggest that in applications where very large amount of data are present, limited computation capabilities and time restrictions, CNN would provide a sufficient solution to aid the process of NID.

Moreover, it is worth noting that during the implementation phase of the experiments, it was observed that the process of training the models required a very high computational time on both workstations in order to complete 1000 epochs in the training phase. In order to tackle the problem of high computation time, a change was made to the batch size of the models from 32 to 64 which yielded a balance in the performance and aided in speeding up the training process. Additionally, although we increased the number of layers in each of the deep learning models used in order to improve the performance, we didn't observe any distinguishable improvement in terms of the accuracy, precision, recall or F1 score.

VI. CONCLUSION AND FUTURE WORK

Currently, Network Intrusion Detection Systems (NIDS) are being discussed heavily in the literature. Generally, there are two broad categories of intrusion detection techniques; one is based on a clear set of rules, technically known as 'signature detection' and other is based on behaviour, technically known as 'anomaly based detection'. The current study applied the later technique of anomaly based intrusion detection using CNN and RNN with two types LSTM and GRU using the NSL-KDD dataset for the purpose of training and testing.

Among the tested deep learning techniques, CNN found to have outperform the other techniques with accuracy, F1 score, recall and precision of above 97%. One of the main limitations that were faced during the research was the limited computational resources that caused longer training time for each model. In order to tackle this issue, we plan to deploy faster systems that are able to train the algorithms in much shorter time. In future work, we will also further investigate the performance of different deep learning techniques by implementing various combinations of deep learning architectures with different depths and hyperparameters and whether these factors will have a great effect on the overall performance of the NIDS.

Acknowledgement

We thank Prof.Amr Mohamed, Professor of Computer Engineering at Qatar University who provided his insights and support throughout this research.

References

[1] D. E. Denning, "An intrusion-detection model," *IEEE Transactions on Software Engineering*, vol. SE-13, no. 2, pp. 222–232, Feb. 1987, ISSN: 0098-5589. DOI: 10.1109/TSE.1987.232894.

[2] M. K. Asif, T. A. Khan, T. A. Taj, U. Naeem, and S. Yakoob, "Network intrusion detection and its strategic importance," in *2013 IEEE Business Engineering and Industrial Applications Colloquium (BEIAC)*, Apr. 2013, pp. 140–144. DOI: 10.1109/BEIAC.2013.6560100.

[3] M. U. Oney and S. Peker, "The use of artificial neural networks in network intrusion detection: A systematic review," *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, 2018. DOI: 10.1109/idap.2018.8620746.

[4] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Applying convolutional neural network for network intrusion detection," *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 2017. DOI: 10.1109/icacci.2017.8126009.

[5] M. Sheikhan, Z. Jadidi, and A. Farrokhi, "Intrusion detection using reduced-size rnn based on feature grouping," *Neural Computing and Applications*, vol. 21, no. 6, pp. 1185–1190, 2010. DOI: 10.1007/s00521-010-0487-0.

[6] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep recurrent neural network for intrusion detection in sdn-based networks," in *2018 4th IEEE Conference on Network Softwarization and Workshops (NetSoft)*, Jun. 2018, pp. 202–206. DOI: 10.1109/NETSOFT.2018.8460090.

[7] G. Zhao, C. Zhang, and L. Zheng, "Intrusion detection using deep belief network and probabilistic neural network," in *2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)*, vol. 1, Jul. 2017, pp. 639–642. DOI: 10.1109/CSE-EUC.2017.119.

[8] C. Li, J. Wang, and X. Ye, "Using a recurrent neural network and restricted boltzmann machines for malicious traffic detection," *NeuroQuantology*, vol. 16, no. 5, 2018, ISSN: 1303-5150. [Online]. Available: https://www.neuroquantology.com/index.php/journal/article/view/1391.

[9] W. Lin, H. Lin, P. Wang, B. Wu, and J. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," in *2018 IEEE International Conference on Applied System Invention (ICASI)*, Apr. 2018, pp. 1107–1110. DOI: 10.1109/ICASI.2018.8394474.

[10] 2019. [Online]. Available: https://github.com/Anihilakos/RNN-LSTM-Network-Intrusion.

[11] Ü. Çavuşoğlu, "A new hybrid approach for intrusion detection using machine learning methods," *Applied Intelligence*, 2019. DOI: 10.1007/s10489-018-01408-x.

[12] L. Dhanabal and D. S. P. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," 2015.

[13] 2019. [Online]. Available: https://www.unb.ca/cic/datasets/nsl.html.

[14] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Jul. 2009, pp. 1–6. DOI: 10.1109/CISDA.2009.5356528.

[15] 2019. [Online]. Available: https://github.com/vinayakumarr/Network-Intrusion-Detection/tree/master/NSL-KDD.