International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835 Volume-4, Issue-4, Aprl.-2017
http://iraj.in

AN EFFICIENT SUPERVISED MODEL FOR INTRUSION DETECTION

[1]THI-THU-HUONG LE, [2]HOWON KIM

[1,2]School of Computer Science and Engineering, Pusan National University, South Korea
E-mail: [1]lehuong7885@gmail.com, [2]howonkim@pusan.ac.kr

Abstract- An Intrusion Detection System (IDS) is a device or software application that monitors a network or systems for malicious activity. In this paper, we consider deep learning is a new approach in this field. The main contributions of this paper are as follows. Firstly, we proposed a supervised model, GRU+BN+Dropout, to detect intrusion. The architecture of this model includes three main layers such as GRU hidden layer, Batch Normalization (BN) layer, and Dropout layer. Secondly, we constructed a learning algorithm of the proposed model. Finally, we have implemented our model and then evaluated its performance classification using several of methods such as confusion matrix, F-measure, and ROC curve. Our model achieved 97% of ROC curve and 94% of F-measure.

Keywords- Deep learning, Gate Recurrent Unit, Intrusion Detection System, Batch Normalization, Dropout, KDD Cup' 99.

I. INTRODUCTION

IDS is a tool to detect activities, which are normal or attack as well as known or unknown malicious in the network. IDS consists of two techniques such as misuse detection (or signature-based) and anomaly detection [1]. Signature-based detection is very effective at detecting known threats [2]. These known patterns referred to as signatures. Anomaly detection is the process of comparing definitions of what activity considered normal against observed to identify significant deviations [2]. These techniques are two different ways of spotting intrusions. Signatures based IDSs rely on humans to create, test, and deploy the signatures. Therefore, it may take hours or days to generate a new signature for an attack, which can be too long when dealing with rapid attacks. Therefore, it is unable to detect unknown attacks [3]. Otherwise, anomaly-based IDSs require storage of normal usage behavior and operate upon audit data generated by the operating system.

From literature survey, we classify into three groups of machine learning used in IDS. They are single classifiers, hybrid classifiers, and ensemble classifiers.

In single classifiers, some researcher who applied a single algorithm in machine learning to detect the anomaly. They are Fuzzy Logic [4], Genetic Algorithms [5], Self-Organizing Maps [6], K-Nearest Neighbor (KNN) [7], Support Vector Machine [8], Artificial Neural Networks [9], etc. Although this group is easy to perform in IDS, the published results are rather not high performance classification. In other words, we should necessary improve this method to enhance performance.

In hybrid classifiers, this group is done to build a better system via combination a few machine learning algorithms. Hybrid architecture is designed and proved that they can improve the performance [10]. Modeling intrusion detection system using hierarchical hybrid intelligent combining decision tree and support vector machine (DT-SVM) was proposed by [11]. DT-SVM produces high detection rate while reduces differentiate attacks from normal behavior. However, this method involves to computing complex computation. In other words, this method gives less reasonable computation in IDS.

In ensemble classifier, this group used to improve the performance of single classification [12]. Ensemble classifier combines the weak of single classifiers and collectively produce better results [13]. However, this method needs more computation cost and hard to implement.

In this study, we investigate the application to network intrusion detection by proposed new model based on conventional Recurrent Neural Network model (RNN). We build the different potential algorithms and later on using this knowledge to develop an intrusion detection classifier as efficiency. We choose the publicly available DARPA (KDD Cup'99) dataset for our experiments because of some reasons as follows. The first, the training data available in IDS for machine learning is very limited. There are two labeled datasets such as KDD Cup'99 datasets [14] and the UNB ISCX 2012 as presented in [15]. The ISCX 2012 was collected more recently in 2010 and was intended as a replacement for the KDD Cup'99.

Unfortunately, availability of the dataset is discontinued. The second, the KDD Cup'99 datasets is still the most comprehensive and widely used for researchers. There are many machine learning algorithms used this dataset to train. The final, the KDD Cup dataset was mainly chosen for the reason of comparison with them. The product of this work is a system which implements a supervised neural network. The network can be trained and tested using the dataset is mentioned above. In order to prove the efficiency of the system, some tests will be carried out the evaluation in the experiment.

This paper is structured in the as follows: Section 2 provides background related our works such as Gate

Recurrent Unit, Batch Normalization, and Dropout. Section 3 describes the architecture of proposed model. Section 4 presents provides the experimental results and comparison to baseline models. Final, we concludes our work.

II. BACKGROUND

Deep learning is a branch of machine learning methods based on learning representations of data. Besides, deep learning can train and learn from high dimensional spaces of data. Recurrent Neural Network (RNN) is one of a famous model in deep learning. RNN is an extension from Feed Forward Neural Network. RNNs are called recurrent because they perform the same task for every element of a sequence and the output being depended on the previous computations. Conventional RNN model used BackPropagation Training Time (BPTT) algorithm to train dataset. However, traditional RNN model gives two limitations. First one is its model comprised difficult to train due to cannot capture long-term dependency. It causes vanishing or exploring gradient descent. The second one is this model may happen overfitting problem while training data. To address the first problem, some previous works proposed the method such as Long Short-Term Memory (LSTM) by Hochrieter and Schiemidbuher [16] and Gated Recurrent Unit (GRU) by Cho et al. [17]. To overcome the second problem, here are some methods to avoid overfitting, including regularization (L2 and L1), Max norm constraints and Dropout. In this work, we approach as a different way to address the first problem. Besides, we choose Dropout method to avoid overfitting for our model.

2.1. Gate Recurrent Unit

Cho et al. proposed the method, GRU, to overcome vanishing gradient problem. In hidden layer, he replaced hidden node by GRU node. Each GRU node consists of two gates that are described in Fig. 1, update gate zt and reset gate rt. Update gate decides how much unit updates its activation or content. And reset gate allows to forget previously computed states.

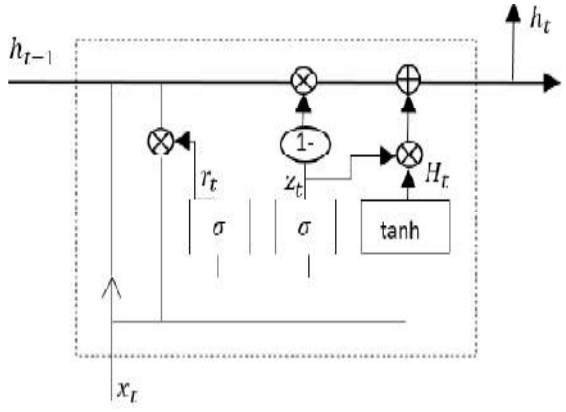

Fig.1. Gated Recurrent Unit

In the GRU, we use model parameters including bias and weigh matrices. bh is the bias at hidden node. Weight matrices of update gate, reset gate, and hidden state are Wz, Wr, WH, respectively. We need to calculate at these gates following bellow equations:

$$z_t = \sigma(W_z * x_t + W_z * h_{t-1}) \quad (1)$$

$$r_t = \sigma(W_r * x_t + W_r * h_{t-1}) \quad (2)$$

$$H_t = tanh\big(W_H * x_t + W_H * (r_t * h_{t-1})\big) \quad (3)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * H_t + b_h \quad (4)$$

2.2. Batch Normalization

Sergey I. at el. [18] proposed Batch Normalization (BN). BN allows us to use much higher learning rates and be less careful about initialization. BN consist of one more steps, which makes this algorithm powerful. Here is BN algorithm:

Input: Value of x over a mini-batch: $B = \{x_1 \dots m\}$;
Parameter to be learned: γ, β
Output: $\{y_i = BN_{\gamma,\beta}(x_i)\}$
Procedure:

$\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ (5) //mini-batch mean

$\sigma_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_B)^2$ (6) // mini-batch variance

$\widehat{x_i} \leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$ (7) // normalize

$y_i \leftarrow \gamma \widehat{x_i} + \beta \equiv BN_{\gamma,\beta}(x_i)$ (8) // scale and shift

2.3. Dropout

Dropout is a technique for addressing this problem. N. Srivastava [19] proposed the technique for dealing with overfitting in neural network. The key idea randomly drops units (along with their connections) from the neural network during training. During training, dropout samples from an exponential number of different "thinned" networks. This significantly reduces overfitting and delicate major improvements over other regularization methods. Dropout in deep learning works as follows: one or more neural network nodes is switched off once in a while so that it will not interact with the network (it weights cannot be updated, nor affect the learning of the other network nodes). With dropout, the learned weights of the nodes become somewhat more insensitive to the weights of the other nodes and learn to decide somewhat more by their own (and less dependent on the other nodes they are connected to). In general, dropout helps the network to generalize better and increase accuracy since the influence of a single node is decreased by dropout.

III. THE PROPOSED MODEL

A proposed classifier for intrusion detection can be constructed on the following steps in Fig. 2. First,

data preprocessing is utilized to data arrangement from original dataset. After that training, the dataset is sent to Proposed Classifier Train to build the learning model. Finally, the system applies Proposed Classifier Test to predict the performance classifier such as Accuracy, Precision, Recall and False Alarm Rate.

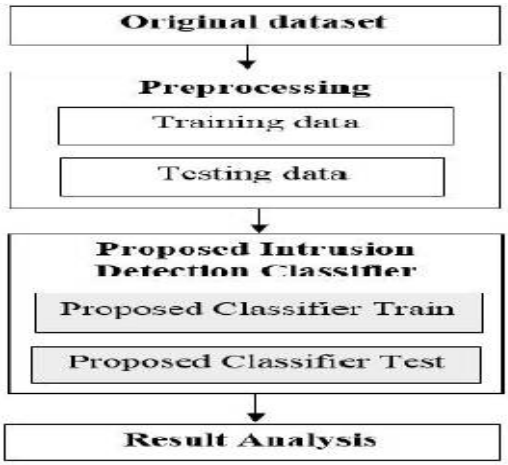

Fig. 2. The Flow of the Proposed Intrusion Detection Classifier

We interest in two factors that are Batch Normalization and Dropout. Batch Normalization is a technique to provide any layer in a Neural Network with inputs that are zero mean/unit variance. Besides, Dropout addresses the overfitting problem when we train conventional RNN model on the large dataset. Our ideas are applied Batch Normalization and Dropout after GRU hidden layer. Here we present the overview structure of the proposed IDS classifier in Fig. 3. After GRU layer, we practice Batch Normalization and then Dropout to overcome overfitting that may occur when we train this model with long-time dependence. Basically, our model consists of three main layers: GRU hidden layer, Batch Normalization (BN) layer, and Dropout layer. To empathize this model, we present the model via unfolding the time (see Fig. 4).

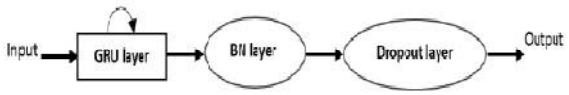

Fig. 3. The proposed intrusion detection classifier

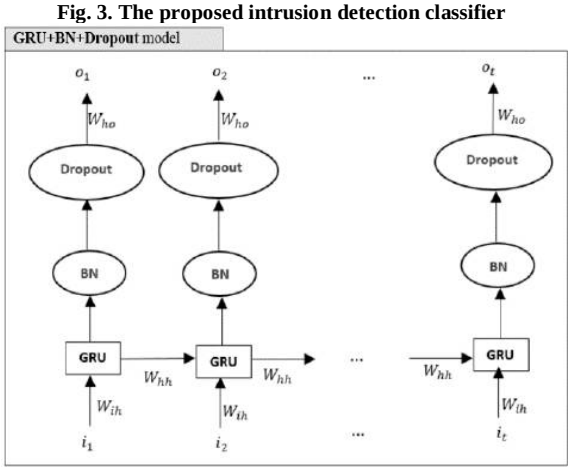

Fig. 4. Unfolding GRU+BN+Dropout classifier by the time

This model consists of five layers. The first one is input layer is denoted by it. The second one is GRU hidden layer is denoted by $h(t)$. Next layer is BN layer is denoted by BN. Dropout layer is the fourth layer. The final layer is output layer is denoted by ot. Table 1 describes some notations in our proposed model.

Table 1: Notations in the proposed model

Notations	Description
W_z	Weight matrix connects between input layer to update gate
W_r	Weight matrix connects between input layer to reset gate
W_{hh}	Recurrent weight matrix in GRU hidden layer
W_{ho}	Weight matrix connects between GRU hidden layer to output layer
b_{hh}	Bias at GRU hidden layer
b_{ho}	Bias at output layer
f	Activation function

Firstly, we necessitate to calculate hidden layer at time t based on the previously hidden state and the input at the current step:

$$z_t = \sigma(W_z * i_t + W_z * h_{t-1}) \quad (9)$$

$$r_t = \sigma(W_r * i_t + W_r * h_{t-1}) \quad (10)$$

$$H_t = tanh(W_{hh} * i_t + W_{hh} * (r_t * h_{t-1})) \quad (11)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * H_t + b_{hh} \quad (12)$$

Secondly, we need to calculate Batch Normalization of ht at time t:

$ht=BatchNormalizat(ht)$ (13)

Thirdly, we require to calculate Dropout of ht at time t:

$ht=Drop(ht)$ (14)

Finally, we take to compute the prediction of the model from GRU hidden layer to output layer.

$ot=(Whoht+bho)$ (15)

IV. EXPERRIMENT

4.1. Dataset

DARPA1998, DARPA1999, and KDD99 are several datasets, which used for IDS classification. KDD99 is mostly used data set. There are many drawbacks of DARPA [20] such as normal attack is not realistic, false alarm behavior cannot be validated. The KDD99 dataset is inherited from DARPA and has got the same limitations. These datasets are publically used and recognized as a standard dataset for IDS. In the real world, KDD99 dataset was used more than the others. This dataset includes connection records with 41 features whose relevance for intrusion detection are not clear. In this dataset, we have four category attacks. They comprise DoS, R2L, U2R, and Probe attacks. Each attack consists of many small attacks. DoS (Denial of Service) is denied legitimate requests to a system. U2R (User-to-Root) is unauthorized access to local superuser (root) privileges. R2L (Remote-to-Local) is unauthorized access from a

remote machine. Probing (Probe) is surveillance and another probing.

4.2. Initialization hyperparameters of the proposed model

The hyperparameters are an important thing to achieve high performance when training the neural network. The neural network consists of two types of hyperparameters such hyperparameters for training and hyperparameter factors of the model. Training hyperparameters include learning rate, the number of each hidden layer, and the number of epochs. Factor hyperparameters comprise advanced activation function, loss function, and optimizer. One important thing is how to choose the suitable value of hyperparameters in our model. In this work, we set up these hyperparameters based on experiences as manually. Table 2 and Table 3 provide these values of hyperparameters in our model.

Table 2: The training hyperparameters of the model

Name of hyperparameter	Value
Learning rate	0.001
Number of each hidden layer	80
Epochs	300

Table 3: The hyperparameter factors of the model

The factors	Name of function
Advanced Activation function	*LeakyReLu()*
Loss function	*Mean Squared Error()*
Optimizer	*Nadam()*

4.3. Evaluation metrics

In our experiment, we use two methods to evaluate our model such as confusion matrix and Receiver Operating Characteristic (ROC) or ROC curve.

First one is confusion matrix to measure performance of our model. There are some metrics to compute such as Accuracy, Precision, Recall, and F-measure. Here are some equations:

$$recall = \frac{TP}{TP + FN} \quad (16)$$

$$precision = \frac{TP}{TP + FN} \quad (17)$$

$$accuracy = \frac{TP + TN}{(TP + FP) + (FN + TN)} \quad (18)$$

F-measure is a measure of a test's accuracy. F-measure considers both the Precision and the Recall of the test to compute the score. Therefore, F-measure is the harmonic mean of precision and recall:

$$F\,measure = 2 \times \frac{precision \times recall}{precision + recall} \quad (19)$$

Where,

TP is a number of predicted as Normal while they actually were Normal.

FP is a number of predicted as Attack while they actually were Normal.

FN is a number of predicted as Normal while they actually were Attack.

TN is a number of predicted as Attack while they actually were Attack.

Besides, we calculate the False Alarm Rate (FAR) which is the ratio of misclassified normal attack.

$$FAR = \frac{FP}{FP + TN} \quad (20)$$

Secondly, we use to ROC curve to evaluate the proposed model. ROC curve is a common evaluation metric for binary classification problems. The curve is created by plotting the true positive rate (TPR) against the false positive rate (FPR) at various threshold settings. If the classifier is very good, the TPR will increase quickly and the Area Under Curve (AUC) will be close to 1. If the classifier is no better than random guessing, the TPR will increase linearly with the FPR and the AUC will be around 0.5. Here we provide the formulas how to calculate TPR and FPR as follows.

$$TPR = recall = \frac{TP}{TP + FN} \quad (21)$$

$$FPR = \frac{FP}{FP + TN} \quad (22)$$

4.4. Experiment results

In this experiment, we evaluate our model by compute F-measure and FAR values. The highest F-measure and the smallest FAR value are, the better model is. Table 4 is a summary these measurement of four models in detail. Obviously, GRU+BN+Dropout model reaches the highest F-measure (93.95%) and the smallest FAR (1.8%) among the baseline models (see Table 4).

Table 4: The F-measure and FAR values of four models

Model	F-measure	FAR
GRU	0.802021417	0.095766
GRU+ Dropout	0.859366125	0.030393
GRU+BN	0.810525	0.04506
GRU+BN+Dropout	**0.939548126**	**0.017994**

In this experiment, we calculate two types of ROC Curve or AUCs. First one is ROC curve for each model. The second one is ROC curve for type of attack. Fig. 5 illustrates ROC cure as well as the ability of performance classification of each model.

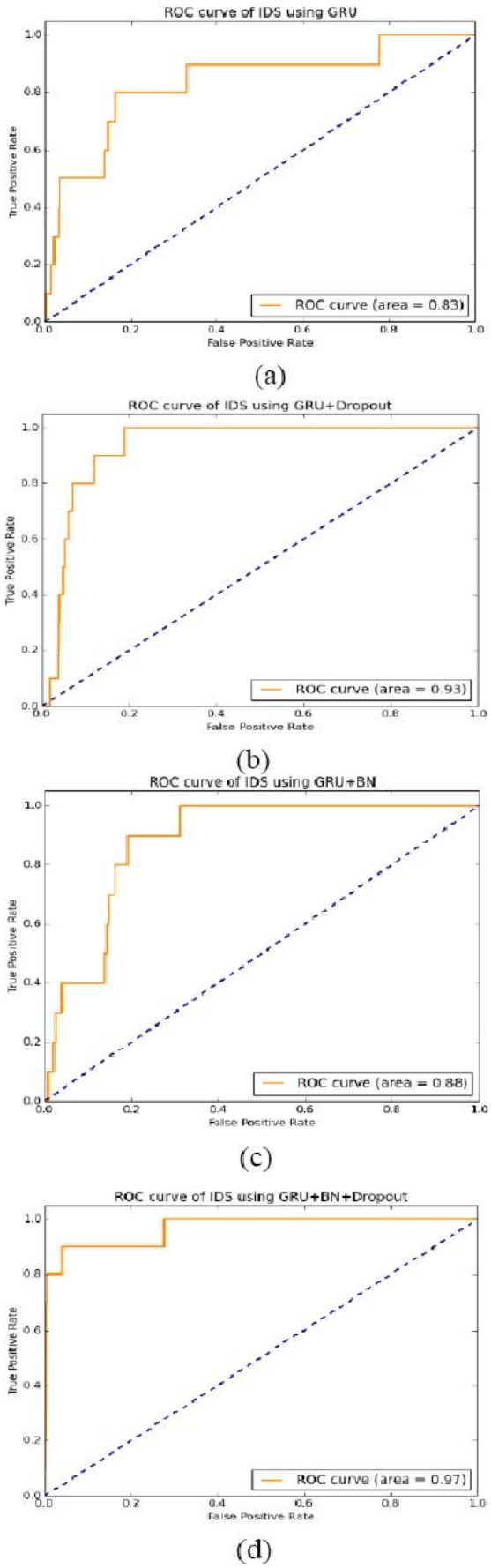

Fig. 5. The ROC curve of four models as follows. (a) GRU model; (b) GRU+ Dropout model; (c) GRU+ BN model; (d) GRU+BN+Dropout model.

CONCLUSIONS

In this paper, we propose the network model of IDS, GRU+BN+Dropout. The architecture of this model consists of three main layers such as GRU layer, BN layer, and Dropout layer. This proposed model can detect intrusion as intimately. We used two measurements to evaluate the proposed model such as confusion matrix and ROC curve.

By our experimentation, the classification performance of our model can reach F-measure at 94%, and only 1.8% of FAR in confusion matrix. In other words, our classifier can achieve high detection rate and low false alarm rate. Besides, our model can obtain 97% of AUC area. The proposed IDS classifier can overcome three existing problems in advanced IDS. The first thing is to detect unknown types of attacks. The second thing is to reduce false positive rate as considerable. Final thing is to avoid the overfitting problem of neural network.

ACKNOWLEDGMENTS

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the Industry4.0s research and development program (S1106-16-1027) supervised by the NIPA (National IT Industry Promotion Agency).

REFERENCES

[1] Lee W and Stolfo S., “Data Mining techniques for intrusion detection”, In: Proc. of the 7th USENIX security symposium, San Antonio, TX, 1998.

[2] Peter Mell, Karen Scarfone, “Guide to intrusion detection and prevention systems (idps)”, National Institute of Standards and Technology, NIST SP - 800-94, 2007. Available at http://www.nist.gov/customcf/get_pdf.cfm?pub_id=50951.

[3] Gandhi, M. and S.K. Srivatsa, “Detecting and preventing attacks using network Intrusion detection systems”. Int. J.Comput.Sci.Sec.,2: pp: 49-60, 2008.

[4] H. Kaur, G. Singh and J. Minhas, “A Review of Machine Learning Based Anomaly Detection Techniques”, International Journal of Computer Applications Technology an d Research, volume 2- issue 2, pp.185-187, 2013.

[5] R. Borgohain, "FuGeIDS: Fuzzy Genetic paradigms in Intrusion Detection Systems", International Journal of Advanced Networking and Applications, vol. 3, no. 6, pp. 1409-1415, 2012.

[6] T. Kohonen, “Self-organized formation of topologically correct feature maps,” Biological Cybernetics, 43, pp.59-69, 1982.

[7] S. Manocha, and M. A. G irolami, “An empirical analysis of the probabilist c K-nearest neighbor classifier”, Pattern Recognition Letters, 28, pp.1818-1824.2007.

[8] S. J. Horng, M. Y. Su, Y. H. Chen, T. W. Kao, R. J. Chen, J. L. Lai and C. D. Kara, "A novel intrusion detection system based on hierarchical clustering and support vector machines", Expert Systems with Applications, 38(1): pp.306-313. 2011.

[9] H. C. Wu and S. H. S. Huang, "Neural networks-based detection of stepping-stone intrusion", Expert Systems with Applications, 37(2): pp.1431 -1437, 2010.

[10] M. Govindarajan and R.M. Chandrasekaran, "Intrusion detection using neural based hybrid classification methods", Computer Networks, 55(8): pp.1662-1671, 2011.

[11] Peddabachigari, S., A. Abraham, C. Grosan and J. Thomas, "Modeling intrusion detection system using hybrid intelligent system", J.Network Comput. Appl., 30: pp.114-131, 2007.

[12] J. Kittler, M. Hatef, R. P. W. Duin and J. Matas, "On combining classifiers", IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(3), pp.226-239, 1998.

[13] F. Majidi, H. Mirzaei, T. Irnapour and F. Faroughi, "A diversity creation method for ensemble based classification: Application in intrusion detection", 2008 7th IEEE International Conference on Cybernetic Intelligent Systems, CIS' 2008.

[14] R. Lippmann, J.Haines, D.Fried, J.Korba and K.Das. "The 1999 DARPA off-line intrusion detection evaluation", Computer network, vol. 34, no.4, pp.579-595, 2000. DOI dx.doi.org/10.1016/S1389-1286 (00)00139-0.

[15] A. Shiravi, H. Shiravi, M. Tavallaee and A.A. Ghorbani. "Toward developing a systematic approach to generate benchmark datasets for intrusion detection", Computer and security, vol.31, no.3, pp.357-474, May 2012. DOI dx.doi.org/10.1016/j.cose.2011.12.012.

[16] S. Hochreiter, J. Dchmidhuber, "Long Short-Term Memory", Neural Computation, pp. 1735-1780 (1997).

[17] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches", arXiv preprint arXiv: pp.1409.1259, 2014.

[18] Sergey I., Christian S., "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", CoRR, 1502.03167, http://arxiv.org/abs/1502.03167, 2015.

[19] N. Srivastava et al., "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", The Journal of Machine Learning Research, Volume 15 Issues 1, pp.1929-1958, Jan 2014.

[20] John McHugh, "Testing Intrusion detection systems: a critique of the 1998 and 1999 DARPA intrusion detection system evaluations as performed by Lincoln Laboratory", ACM Trans. Inf. Syst. Secur., 3(4): pp.262- 294.

★★★