A Multiclass Approach for Network Intrusion Detection using Convolutional Neural Networks

[1]Shashank Shekhar, [2]Abhinav Mittra

[1]Student, [2]Student
Department of Electronics and Communication Engineering,
[2]Department of Computer Science and Engineering
Manipal Institute of Technology, Manipal, Karnataka, India

***Abstract*: The immense popularity of Internet of Things (IoT) and Cloud based applications have resulted in huge volumes of network traffic. Different versions of operating systems, multiple protocols and concurrent users contribute significantly towards the ever increasing computer security threats. Traditional methods involving shallow learning techniques like Random Forest, Naive Bayes, etc. have been instrumental in advancing the study of network intrusion detection. However, as and when the network data expands in size and complexity, deep learning algorithms are required to tackle the ongoing network security challenges. Deep learning methods are intrinsically capable of handling enormous data and their performance increases with increasing supply of the same. The proposed work details the configuration of a multi-class classifier using Convolutional Neural Networks. UNSW NB-15, a modern dataset comprising of nine contemporary attack types is used to evaluate the effectiveness of the proposed approach. Results indicate that the proposed approach has exhibited a reasonably valid precision and recall percentage as compared to the preexisting methods.**

***Index Terms*: Network Intrusion Detection System, Machine Learning, Convolutional Neural Networks, UNSW NB-15**

I. INTRODUCTION

An intrusion detection system is a powerful security mechanism capable of detecting, preventing and reacting to computer invasions [1]. The work related to network intrusion detection dates back to 1987 when Dorothy Denning, while observing network data discovered that the system could send alarms before invasion [1]. The four different approaches to network intrusion detection, namely, anomaly, misuse, hybrid and policy are explained in [2]. Signature based or misuse intrusion detection systems may prove to be ineffective to detect unknown attacks whereas anomaly based systems seem to be effective in detecting unknown vulnerabilities [3]. However, anomaly based intrusion detection systems suffer from large number of false positives. In order to address the limitations of these two types of intrusion detection systems, hybrid intrusion detection systems were proposed that incorporated the features of both signature and anomaly based intrusion detection systems [3].
Network intrusion detection systems are indispensable to computer security because hackers at an alarming rate are compromising the confidentiality, integrity and availability of computer resources.
Multi-class classification problem, when applied in the area of network intrusion detection is technically challenging because majority of the classes are imbalanced. The problem at hand involved the assignment of network instances to exactly one attack type. Given the real world scenario, it is paramount to configure an intrusion detection system that has the capability to distinguish between various attack types. Network traffic originates from heterogeneous sources and it becomes extremely crucial that the proposed models possess an innate property to differentiate between various attack types. In light of the challenges posed in this problem, we have conducted experiments with different machine learning models in order to build a robust system capable of providing results above the baseline on UNSW-NB15, the most widely accepted data set for NIDS problems.

II. RELATED WORK

This section highlights the key contributions made by different authors to solve the problem of network intrusion using various data sets such as KDD CUP 99, DARPA, CDX etc.
A hybrid intrusion detection system was proposed using Self-organizing maps (SOM) and J.48 decision tree [7]. Authors in [8] proposed Efficient Data Adapted Decision Tree (EDADT) algorithm to assist system administrators to analyze the attacks more efficiently and also to address the problem of human interaction involved in pre-processing of enormous network data.
With large-scale advancements in technology, deep learning algorithms are becoming more conspicuous due to their ability to use multiple hidden layers for learning complex patterns. A comprehensive survey was presented in [9], which detailed the application of deep learning architectures in the area of cyber security. Related studies on intrusion detection, spam and malware detection were delineated in the survey. Another survey article investigated the deep learning techniques used for anomaly detection and highlighted the credibility of deep learning algorithms for network traffic analysis [10].
Authors in [11] applied Convolutional Neural Network (CNN) to build an intrusion detection system using KDD cup 99 dataset. They obtained an attack detection percentage of 97.34, 56.26, 61.47, 0 and 94.55 with respect to Denial of service (DOS), Probe, Remote to Local (R2L), User to Root(U2R) and normal classes respectively. A deep learning approach was proposed in [12] to build a classification model using KDD Cup 99 dataset. Authors asserted that their approach exhibited better accuracy as compared to other conventional methods like random forest, artificial neural network and Naive Bayes. A nonsymmetrical deep Auto-Encoder (NDAE)

model was also put forth. Results obtained from the model were quite promising and training time was comparatively lesser than deep belief networks (DBN) [13].
Self-taught Learning (STL), a deep learning technique that was applied on KDD Cup 99 dataset to develop a network intrusion detection system and the authors implemented a stacked Auto-Encoder to perform unsupervised feature learning. This article highlighted the fact that deep learning algorithms are capable of deep packet inspection, which is often required to analyze the incoming packets on the network [14].
Owing to the limitations pertaining to KDD cup 99 dataset like presence of duplicated records and absence of modern attack vectors [16-18], we considered UNSW NB-15 dataset for the proposed study. Founders of UNSW NB-15 dataset conducted a statistical analysis and explained all its attributes. Five classifiers were applied on the dataset to assess its complexity and results were presented in terms of accuracy and false alarm rate. Results indicated that decision trees accomplished the best competence as compared to other classifiers [18]. REPTree, a two stage classification approach was proposed in [19]. Authors were able to achieve 81.28% accuracy but class wise results were not presented. Therefore, it was not possible to comprehend the exact predictions pertaining to a particular attack type found in UNSW NB-15.
A novel method for network intrusion detection was proposed through skip-gram modeling [20]. This technique hypothesized that systems and network connections can be represented as words. Authors emphasized that enormous log files need to be processed for detecting attacks. This work [20], in spite of being novel can be considered as an unsupervised binary classification since the authors did not elucidate the performance of skip-gram model in terms of each attack type. A wrapper approach was applied on UNSW NB-15 dataset using genetic algorithms as a feature selection technique and logistic regression as the classifier [22]. This article is another prominent contribution regarding multiclass classification. Thus, we have compared the class wise results of our proposed approach with this work.

III. DATASET DESCRIPTION

As discussed by the founders [16-18], UNSW NB-15 is a modern dataset consisting of many variants of contemporary attack types. Tools like Argus and Bro-IDS were used in the process of creating this dataset to extract all the possible features and label the records. In order to ensure the authenticity of the present-day attack vectors, attack behavior was envisaged according to the CVE (Common Vulnerabilities and Exposures) website. All the illegitimate network traffic was grouped into nine attack classes namely: Backdoor, Analysis, Fuzzers, Shellcode, Reconnaissance, Exploits, Denial of Service, Worms and Generic [16]. The nine attack categories are defined as follows:
1. Analysis: Intrusions which occur when web applications are penetrated via ports.
2. Backdoor: Security controls are evaded in a mysterious manner to access important information from computers.
3. Denial of Service: Services are made unavailable to legitimate users on the network.
4. Exploits: A software vulnerability is capitalized for gains.
5. Fuzzers: Random data is given as input to detect flaws pertaining to operating systems, network or programs ultimately leading to system crash.
6. Generic: Every possible block-cipher is exploited using multiple hash functions.
7. Worms: A piece of software which multiplies across the network.
8. Reconnaissance: Probing attempts are made to understand the target network better.
9. Shellcode: A segment of code originating from the shell is exploited eventually compromising the target machine.

UNSW NB-15 includes 42 features as described in [16]. The training and testing distributions of the dataset are given below.

Table 1: Number of instances of each attack type in the dataset

Class	**Training Samples**	**Testing Samples**	**Total**
Normal	56000	37000	93000
Analysis	2000	677	2677
Backdoor	1746	583	2329
Reconnaissance	10491	3496	13987
Shellcode	1133	378	1511
Worms	130	44	174
DOS	12264	4089	16353
Fuzzers	18184	6062	24246
Generic	40000	18871	58871
Exploits	33393	11132	44525

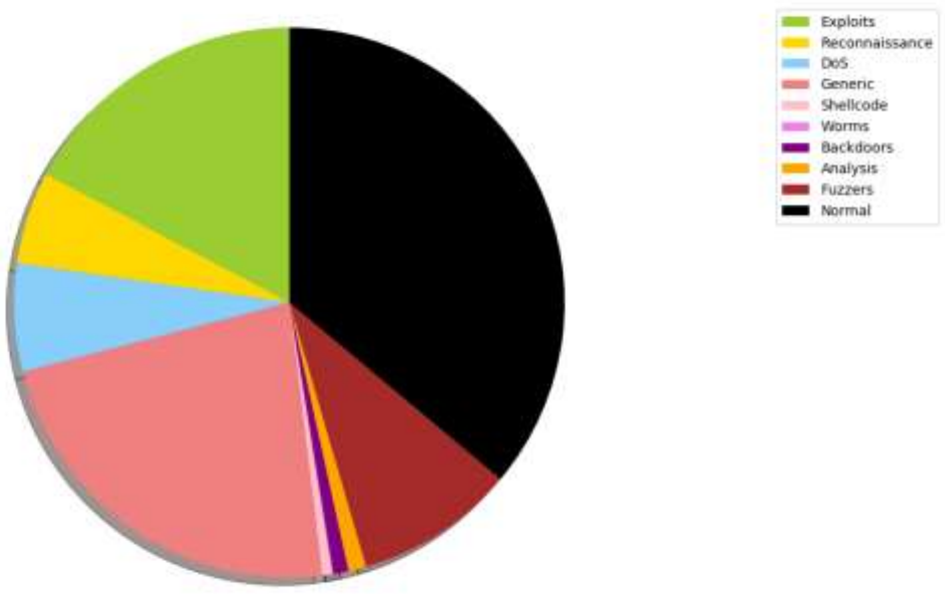


FIGURE 1. Pie Chart Depicting the Distribution of Each Attack Type

IV. METHODOLOGY

A. PRE-PROCESSING

In order to transform the existing dataset into a structure suitable for further analysis, pre-processing was done. The training and testing dataset was combined into a single pandas dataframe. To facilitate appropriate analysis, the four categorical features namely, **proto, service, state and attack cat** were converted into float64 via label encoding. Usually, two methods are predominantly used for data scaling: normalization and standardization. A feature scaling method, **StandardScaler [23]** was employed to scale the features as shown in Equation (1).

$$Z = \frac{x - mean(x)}{Standarddeviation(x)} \qquad (1)$$

The dataset was partitioned into two subclasses, namely the one comprising of class labels and another with target label or the attack type. 80:20 split was used to partition the dataset into training and testing sets. The primary focus of this article is to apprise readers on the proficiency of CNN to characterize the various types of data packets.

B. FEATURE SELECTION

Feature selection is the process of choosing the features, which are most relevant to the target class and are independent of each other. In order to get an in depth knowledge about how the various features were affecting our data, we used Random Forests first followed by Boruta, a wrapper approach to Random Forest available in R. This led us to shortlist the most important features for each category and also observe the consistently high scoring features.

C. RANDOM FOREST

Random Forest classifier was used to select the top 10 features for binary classification. It is a bagging technique which consists of a number of decision trees running together. It is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.
Random decision forests correct for decision trees' habit of overfitting to their training set.

D. BORUTA

Boruta (a wrapper built around the random forest algorithm) feature selection method in R was used to select the top 10 features for each attack type. It tries to capture all the important, interesting features you might have in your dataset with respect to an outcome variable. First, it duplicates the dataset, and shuffles the values in each column. These values are called shadow features. Then, it trains a classifier, such as a Random Forest Classifier, on the dataset. By doing this, it is ensured that you have an idea of the importance via the Mean Decrease Accuracy or Mean Decrease Impurity for each of the features of your data set. The higher the score, the better/important is the feature.
Then, the algorithm checks for each of your real features if they have higher importance. That is, whether the feature has a higher Z-score than the maximum Z-score of its shadow features. If it does, it records this in a vector. This is called a hit. Next, it will continue with another iteration. After a predefined set of iterations, you will end up with a table of these hits.(A Z-score is the number of standard deviations from the mean a data point is).
At every iteration, the algorithm compares the Z-scores of the shuffled copies of the features and the original features to see if the latter performed better than the former. If it does, the algorithm will mark the feature as important. In essence, the algorithm is trying

to validate the importance of the feature by comparing with random shuffled copies, which increases the robustness. This is done by simply comparing the number of times a feature did better with the shadow features using a binomial distribution.
The graphs for the top 10 features have been provided under the Appendix section. The table given below provides a brief look into our findings:

Table 2: Most Important Features for each Attack Type using Boruta

Attack Type	Features
Normal	ct_srv_src, ct_srv_dst, ct_dst_src_ltm
Analysis	ct_src_ltm, ct_srv_src, ct_dst_src_ltm
Backdoor	ct_src_ltm, ct_dst_src_ltm, ct_srv_src
Reconnaissance	rate, dload, ct_src_dport_ltm
Shellcode	sload, sjit, rate
Worms	ct_dst_ltm, ackdat, ct_src_ltm
Denial of Service	ct_src_dport_ltm, sinpkt, dur
Fuzzers	ct_dst_src_ltm, ct_srv_dst, ct_srv_src
Generic	ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm
Exploits	ct_src_ltm, ct_dst_src_ltm, dur

E. EARLY APPROACH
Our initial approach revolved around using Random Forests for feature selection and then, using those features, constructing a single layer feed forward neural network to make accurate classifications. The model performed quite well for binary classification(attack vs normal) but failed for the multiclass classification task (9 attack types and 1 Normal type). This was expected since our architecture was too simple to accurately model the high dimensional data.
At this point, we started exploring the idea of using all the features and modeling the data in a manner that it may be suitable for some deep learning architectures primarily used in image processing.

F. CNN APPROACH
Firstly, Convolutional Neural Networks were chosen for multiclass classification due to their low computational cost and extreme robustness.
A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a CNN is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, CNNs have the ability to learn these filters/characteristics.
The CNN architecture we settled with consists of 6 hidden layers along with the input and output(softmax of 10 neurons) layers. In order to implement and train a customized CNN, we used Conv2D class available in Keras. The Conv2D constructor has three important parameters namely: filters, kernel size and strides. The deeper layers in the network learn more filters whereas early layers learn less filters. Kernel size determines the height and width of the 2D convolution window. Strides indicate the step of the convolution along x and y axis.
After each convolutional block, leaky ReLu, max pooling and dropout of varying values were applied. Leaky ReLu was used since it prevents the neurons from "dying" which might happen with conventional activations due to vanishing gradient problem. Max pooling reduces the computational cost since it just takes the maximum value from a pre-specified kernel for low level feature abstraction. Dropout has been found out by various studies to be extremely effective in minimizing overfitting. Various values for the dropout were experimented upon, starting from 0.5 and gradually tuning down till the optimum set of values was reached.
Final softmax layer was used to assign the probabilities for prediction so it needs to be the same size as the number of outputs (10 in our case). All the convolutional blocks were of 3x3 dimension followed by a max pooling block of 2x2 dimension. Activation function for the convolutional layers was linear. The loss was categorical cross entropy since we were dealing with a multi class classification task and the optimizer used was Adam. The training weights were saved into an hdf5 file so that successive hyper-parameter tuning cycles could be expedited.
It was found that among various values, a satisfactory tradeoff between performance and computational time was achieved by setting the batch size as 256 and the number of epochs as 200.
Finally, the evaluation on the testing set was done after completion of training and the accuracy and loss for each epoch was closely monitored. A confusion matrix was obtained after testing and various performance metrics such as recall, precision, F1 score, FAR, FNR etc were calculated and compared with the benchmark paper.

G. RESNET APPROACH
Since our results using CNN were really encouraging, it was decided to try ResNets too in order to take the research a step further.
The entire data was preprocessed and encoded as explained in the CNN report. Separate identity and convolutional blocks were created. It was expected that the skip connections with identity mapping applied in deep neural networks to remove the problem of vanishing and exploding gradients would be helpful in obtaining satisfactory results. The training and testing set was taken as it was

provided in the dataset folder. Shortcut values were added to the main path and passed through a ReLu activation function. After this, the blocks were stacked together in a ResNet50 architecture. The resulting variable was then flattened and then passed into the softmax layer to make useful predictions. The model was compiled and the results were evaluated on the testing set.
The results were almost same as that obtained on the CNN. A possible cause for this might be the highly skewed distribution of the data among the training and testing set with the testing set being nearly twice as large as the training set which is bound to produce the extremely erroneous results.

V. RESULTS

This section highlights the key observations as per the results obtained by configuring CNN model. The model summary of the CNN used to propose the multiclass approach as mentioned in methodology. We have considered three performance metrics to validate the effectiveness of the proposed approach namely: Precision, Recall and F1-score. Equations (2), (3) and (4) define the three performance metrics.

$$Precision = \frac{TP}{TP+FP} \qquad (2)$$

$$Recall = \frac{TP}{TP+FN} \qquad (3)$$

$$F1 - score = 2\frac{PrecisionXRecall}{Precision+Recall} \qquad (4)$$

We can note that [21] and [22] have also presented class wise predictions in terms of only recall and precision. F1-score is like an umbrella term, which incorporates both precision and recall (F1-score is the harmonic mean between precision and recall). There is always a visible tradeoff between precision and recall. As a more meaningful measure of model's consistency, F1-score is normally preferred.

The confusion matrix shown below is a comparison between actual (A) versus predicted (P) classes. A summary of prediction results pertaining to 9 attack types namely: Backdoor(B), Analysis(A), Fuzzers(F), Exploits(E), Reconnaissance(R), Denial of Service(D), Shellcode(S), Worms(W) and Generic(G) and Normal (N) can be ascertained from the confusion matrix.

Table 3: Confusion Matrix

A	**N**	**B**	**A**	**F**	**E**	**R**	**D**	**S**	**W**	**G**	**Recall %**
N	**17383**	0	2	872	101	108	1	22	0	0	94
B	8	**7**	0	2	412	10	0	9	0	0	2
A	96	0	**27**	2	412	0	1	0	0	0	5
F	1674	0	0	**2477**	624	79	2	25	0	0	51
E	312	2	1	136	**8257**	270	36	51	0	3	91
R	39	0	0	11	545	**2134**	10	4	0	0	78
D	65	0	0	34	2952	32	**115**	40	0	0	4
S	34	0	0	19	37	34	0	**170**	0	0	58
W	2	0	0	2	24	0	0	2	**2**	0	6
G	17	0	0	37	238	2	28	7	0	**0**	97
Precision (%)	89	78	90	69	61	80	60	52	100	100	

F1-scores of all the ten classes are tabulated in the table below:

Table 4: F1 Scores for all Attack Types

Class/Category	F1 Score (%)
Normal	91
Backdoor	3
Analysis	10
Fuzzers	58
Exploits	73
Reconnaissance	79
Denial of Service	7
Shellcode	54
Worms	12
Generic	99

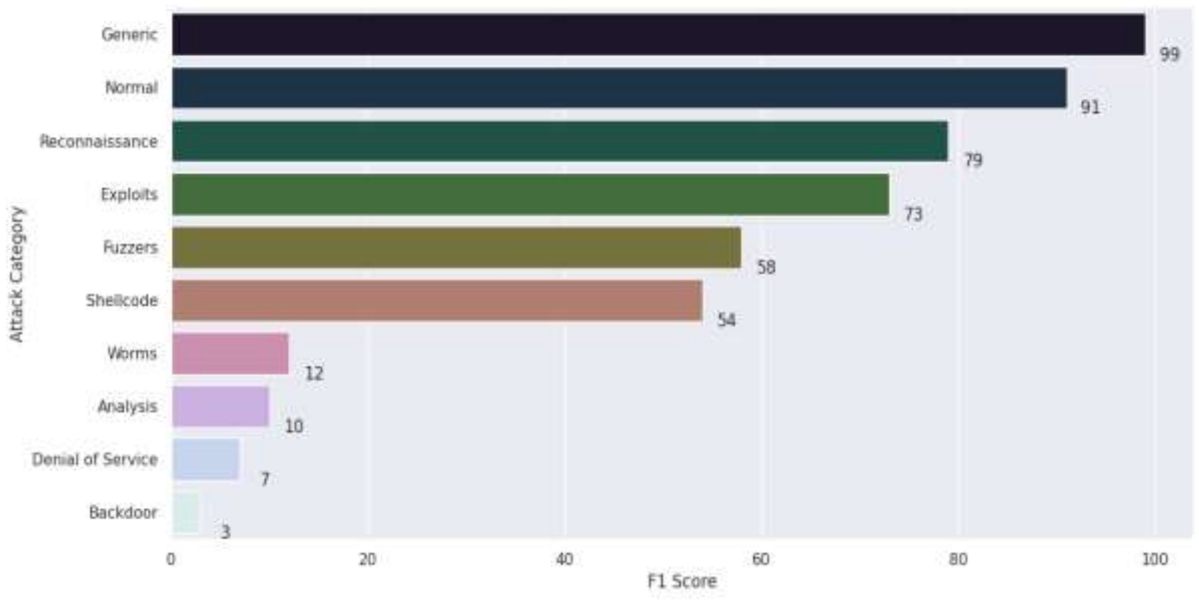


FIGURE 2: F1 Scores of all Attack Types

VI. DISCUSSION

In this section, we have compared the performance of our proposed approach with two state of the art methods [21] and [22]. It is a matter of general understanding that there cannot be any technique or algorithm capable of detecting all the network instances of a class perfectly. The rationale behind this fact is probably the considerable tradeoff which often occurs with respect to recall and precision while handling imbalanced datasets. As mentioned earlier, Dendron and GA-LR are the two state of the art methods demonstrating multiclass classification on UNSW NB-15 dataset. Hence we have compared the proposed model's performance with the two above said methods as summarized below.

i. Dendron: The proposed multiclass approach has exhibited a fairly good recall percentage with respect to four classes namely Shellcode, Reconnaissance, Exploits and Generic as compared to [21]. It can be noted that recall percentage of Shellcode, Reconnaissance, Exploits and Generic classes are 58%, 78%, 91% and 97% respectively as achieved by the proposed approach in comparison to 36.39%, 46.07%, 76.22% and 81.37% as reported in [21].

ii. Dendron: The performance of the proposed approach has been superior in terms of precision pertaining to seven classes particularly Analysis (90%), Shellcode (52%), Reconnaissance (80%), Backdoor (78%), Worms (100%), Generic (100%) and Denial of service (60%). It is worthwhile to mention that [21] reported a precision percentage of 68.33%, 16.1%, 63.55%, 17.68%, 4%, 98.1% and 20.26% corresponding to the aforementioned classes.

iii. GA-LR: Another prominent article [22] depicted a clear class-wise performance of all the ten classes. We have compared our proposed multiclass approach with [22] also. Three classes like Normal (94%), Shellcode (58%) and Reconnaissance (78%) have reported a higher recall percentage than [22] as observable in the proposed approach. It can be noted that [22] reported 90.7%, 47.4% and 76% recall percentage with respect to the above three mentioned classes.

iv. GA-LR: From the results of the proposed approach, it is evident that six classes such as Backdoor (78%), Analysis (90%), Exploits (61%), Denial of Service (60%), Worms (100%) and Generic (100%) have reported a higher precision percentage than [22]. It was noticed that [22] reported 51.5%, 44.44%, 60.20%, 36.09%, 46.875% and 99.731% as precision with regard to the afore mentioned classes.

VII. CONCLUSION

The proposed multiclass approach detailed the configuration of a convolutional neural network model. As explained in the preceding section, it is seldom possible to obtain a conclusive recall and precision percentage owing to imbalanced datasets. We recapitulate that multiclass classification problem is far more coercive in the study of network intrusion detection systems than binary classification. This is applicable mainly because the IDS should possess the capability to discern between various attack types. Thus, we devised a CNN model to achieve the above said objective. We conducted a study on the available methods for multi-class classification and attempted a class wise comparison between the proposed approach and the existing techniques. Our inference points out that the proposed approach performs on par with the current approaches barring a few trade-offs. As part of future work, we aim to operate on disparate intrusion detection datasets and test the efficacy of our proposed methodology.

REFERENCES

[1] Denning, Dorothy E. "An intrusion-detection model." IEEE Transactions on software engineering 2 (1987): 222-232.

[2] Agarwal, Nancy, and Syed Zeeshan Hussain. "A closer look at Intrusion Detection System for web applications." Security and Communication Networks 2018 (2018).

[3] Liao, Hung-Jen, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. "Intrusion detection system: A comprehensive review." Journal of Network and Computer Applications 36, no. 1 (2013): 16-24.

[4] Tsai, Chih-Fong, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin. "Intrusion detection by machine learning: A review." expert systems with applications 36, no. 10 (2009): 11994-12000.

[5] Kumar, Gulshan, and Krishan Kumar. "The use of artificial-intelligence-based ensembles for intrusion detection: a review." Applied Computational Intelligence and Soft Computing 2012 (2012): 21.

[6] Han, Sang-Jun, and Sung-Bae Cho. "Detecting intrusion with rule-based integration of multiple models." Computers & Security 22, no. 7 (2003): 613-623.

[7] Depren, Ozgur, Murat Topallar, Emin Anarim, and M. Kemal Ciliz. "An intelligent intrusion detection system (IDS) for anomaly and misuse detection in computer networks." Expert systems with Applications 29, no. 4 (2005): 713-722.

[8] Nadiammai, G. V., and M. Hemalatha. "Effective approach toward Intrusion Detection System using data mining techniques." Egyptian Informatics Journal 15, no. 1 (2014): 37-50.

[9] Mahdavifar, Samaneh, and Ali A. Ghorbani. "Application of Deep Learning to Cybersecurity: A Survey." *Neurocomputing* (2019).

[10] Kwon, Donghwoon, Hyunjoo Kim, Jinoh Kim, Sang C. Suh, Ikkyun Kim, and Kuinam J. Kim. "A survey of deep learning-based network anomaly detection." Cluster Computing (2017): 1-13.

[11] Liu, Yuchen, Shengli Liu, and Xing Zhao. "Intrusion detection algorithm based on convolutional neural network." DEStech Transactions on Engineering and Technology Research iceta (2017).

[12] Yin, Chuanlong, Yuefei Zhu, Jinlong Fei, and Xinzheng He. "A deep learning approach for intrusion detection using recurrent neural networks." Ieee Access 5 (2017): 21954-21961.

[13] Shone, Nathan, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. "A deep learning approach to network intrusion detection." IEEE Transactions on Emerging Topics in Computational Intelligence 2, no. 1 (2018): 41-50.

[14] Javaid, Ahmad, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. "A deep learning approach for network intrusion detection system." In Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), pp. 21-26. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016.

[15] Potluri, Sasanka, and Christian Diedrich. "Accelerated deep neural networks for enhanced intrusion detection system." In 2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8. IEEE, 2016.

[16] Moustafa, Nour, and Jill Slay. "The significant features of the UNSW-NB15 and the KDD99 data sets for network intrusion detection systems." In 2015 4th international workshop on building analysis datasets and gathering experience returns for security (BADGERS), pp. 25-31. IEEE, 2015.

[17] Moustafa, Nour, and Jill Slay. "UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)." In 2015 military communications and information systems conference (MilCIS), pp. 1-6. IEEE, 2015.

[18] Moustafa, Nour, and Jill Slay. "The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set." Information Security Journal: A Global Perspective 25, no. 1-3 (2016): 18-31.

[19] Belouch, Mustapha, Salah El Hadaj, and Mohamed Idhammad. "A two-stage classifier approach using reptree algorithm for network intrusion detection." International Journal of Advanced Computer Science and Applications (ijacsa) 8, no. 6 (2017): 389-394.

[20] Carrasco, Rafael San Miguel, and Miguel-Angel Sicilia. "Unsupervised intrusion detection through skip-gram models of network behavior." Computers & Security 78 (2018): 187-197.

[21] Papamartzivanos, Dimitrios, Felix Gomez Marmol, and Georgios Kambourakis. "Dendron: Genetic trees driven rule induction for network intrusion detection systems." Future Generation Computer Systems 79 (2018): 558-574.
[22] Khammassi, Chaouki, and Saoussen Krichen. "A GA-LR wrapper approach for feature selection in network intrusion detection." computers & security 70 (2017): 255-277.
[23] Buitinck, Lars, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae et al. "API design for machine learning software: experiences from the scikit-learn project." arXiv preprint arXiv:1309.0238 (2013).
[24] O'Shea, Keiron, and Ryan Nash. "An introduction to convolutional neural networks." arXiv preprint arXiv:1511.08458 (2015).
[25] Wu, Jianxin. "Introduction to convolutional neural networks." National Key Lab for Novel Software Technology. Nanjing University. China (2017): 5-23.
[26] Stutz, David. "Understanding convolutional neural networks." InSeminar Report, Fakultät für Mathematik, Informatik und Naturwissenschaften Lehr-und Forschungsgebiet Informatik VIII Computer Vision (2014).
[27] Koushik, Jayanth. "Understanding convolutional neural networks." arXiv preprint arXiv:1605.09081 (2016).
[28] Ketkar, Nikhil. "Introduction to keras." In Deep Learning with Python, pp. 97-111. Apress, Berkeley, CA, 2017.
[29] Xu, Bing, Naiyan Wang, Tianqi Chen, and Mu Li. "Empirical evaluation of rectified activations in convolutional network." arXiv preprint arXiv:1505.00853 (2015).
[30] Maas, Andrew L., Awni Y. Hannun, and Andrew Y. Ng. "Rectifier nonlinearities improve neural network acoustic models." In Proc. icml, vol. 30, no. 1, p. 3. 2013.
[31] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." arXiv preprint arXiv:1412.6980 (2014).

APPENDIX

A) GRAPHS FOR TOP FEATURES OF EACH ATTACK TYPE

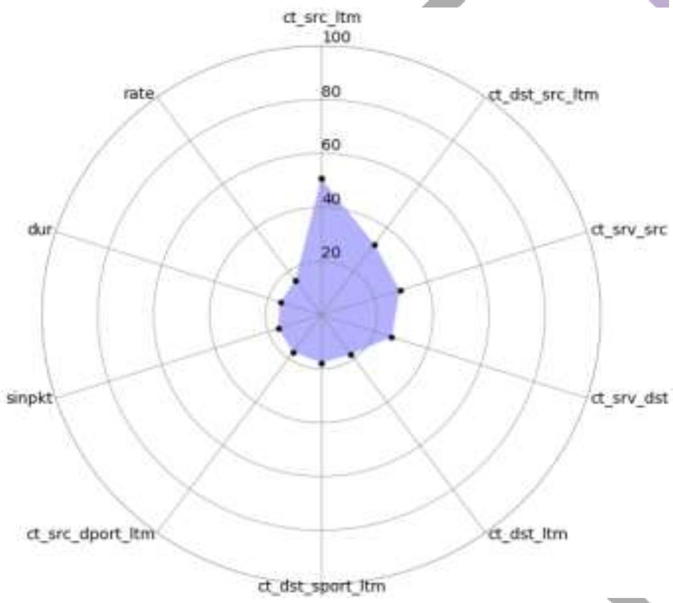


FIGURE 3: ANALYSIS

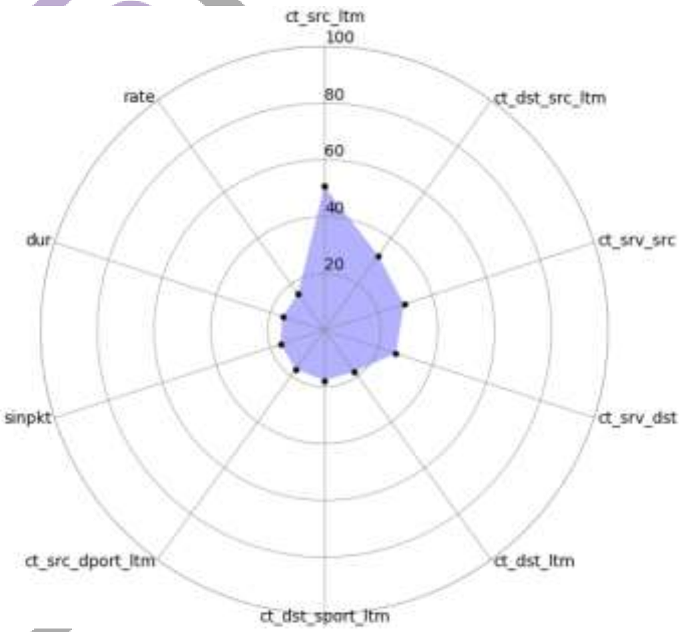


FIGURE 4: BACKDOOR

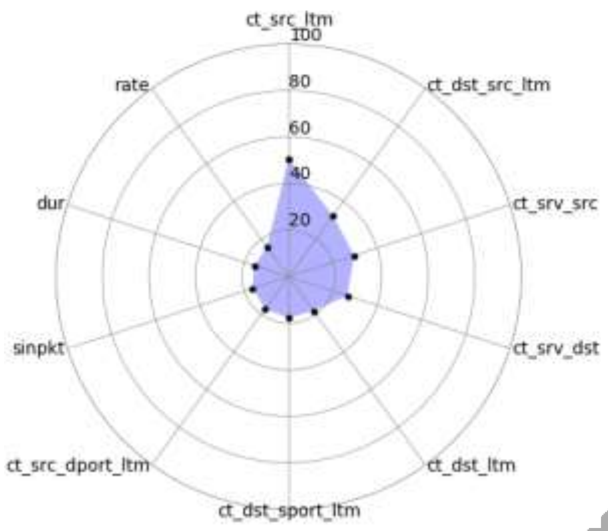


FIGURE 5: DENIAL OF SERVICE

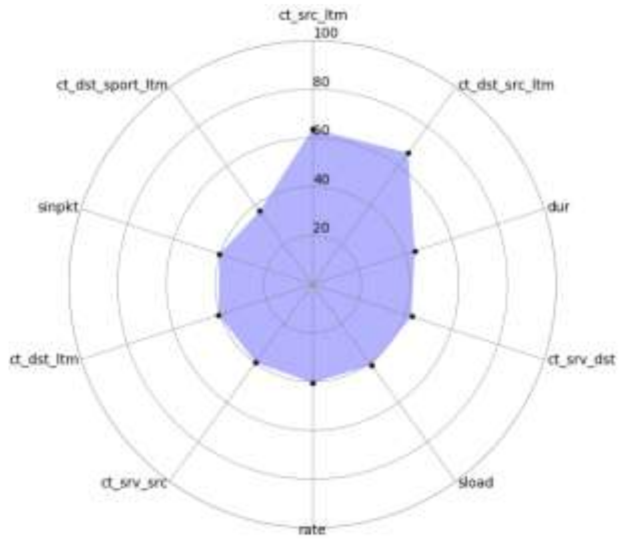


FIGURE 6: EXPLOITS

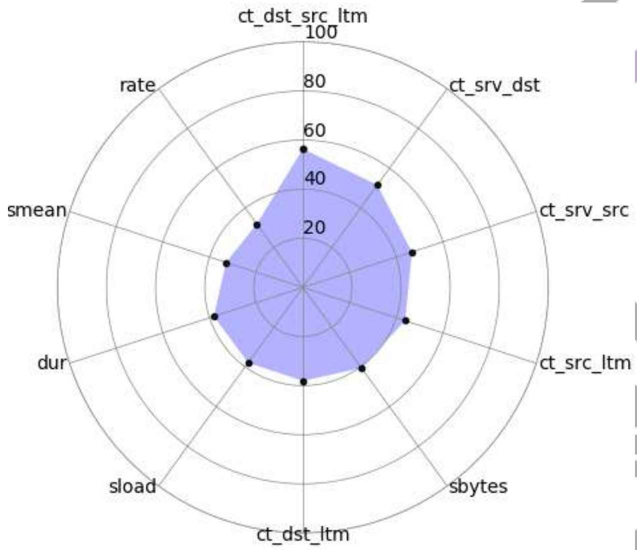


FIGURE 7: FUZZERS

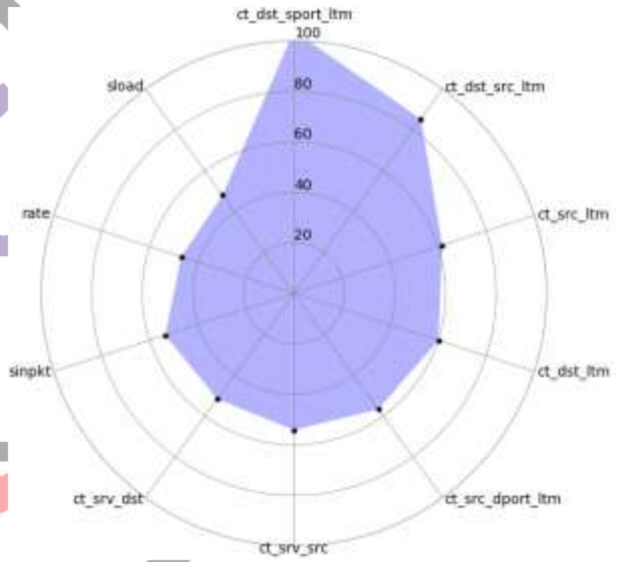


FIGURE 8: GENERIC

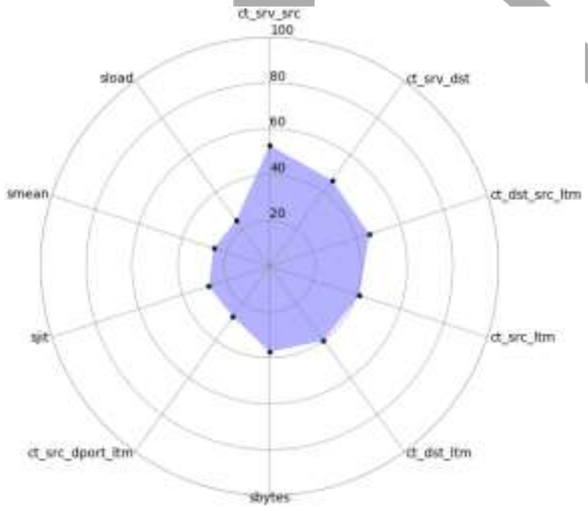


FIGURE 9: NORMAL

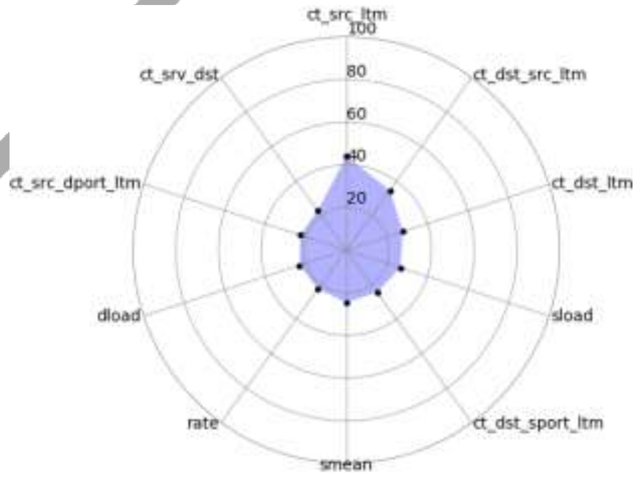


FIGURE 10: RECONNAISSANCE

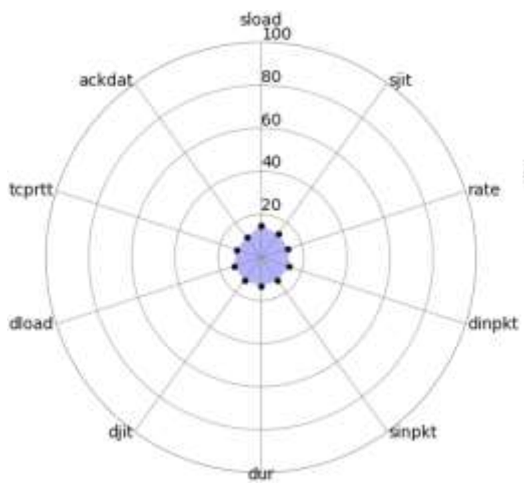


FIGURE 11: SHELLCODE

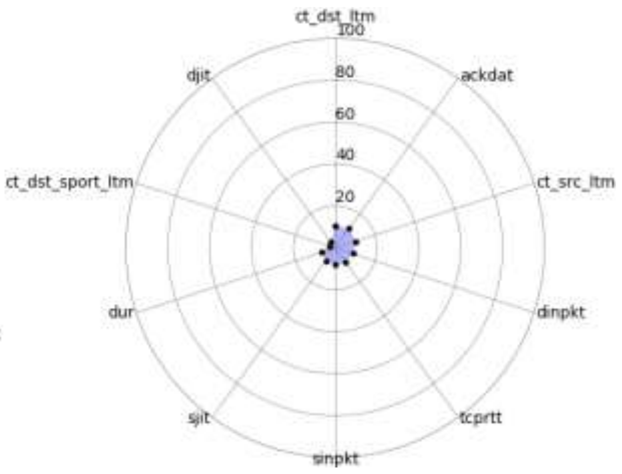
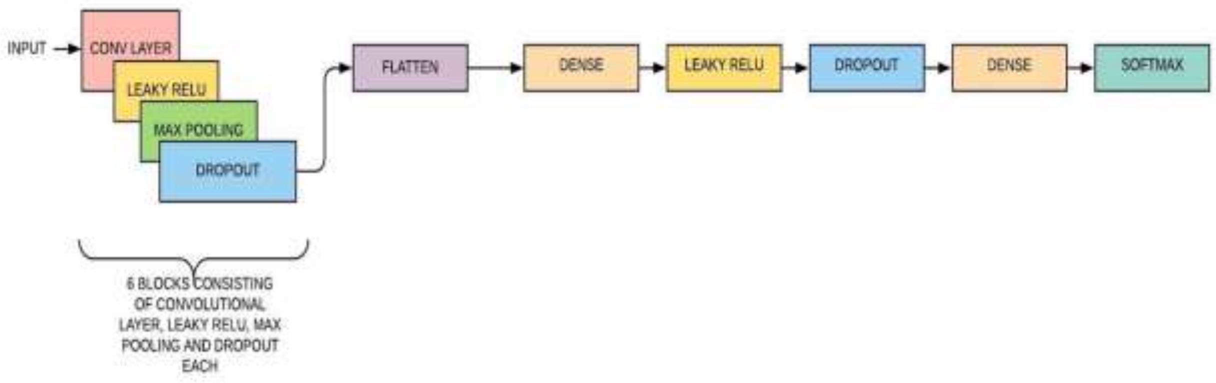


FIGURE 12: WORMS

B) **BLOCK DIAGRAM FOR THE CONVOLUTIONAL NEURAL NETWORK MODEL ARCHITECTURE**


INPUT
CONV LAYER
LEAKY RELU
MAX POOLING
DROPOUT
FLATTEN
DENSE
LEAKY RELU
DROPOUT
DENSE
SOFTMAX
6 BLOCKS CONSISTING OF CONVOLUTIONAL LAYER, LEAKY RELU, MAX POOLING AND DROPOUT EACH